PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2003-173275
(43)Date of publication of application : 20.06.2003

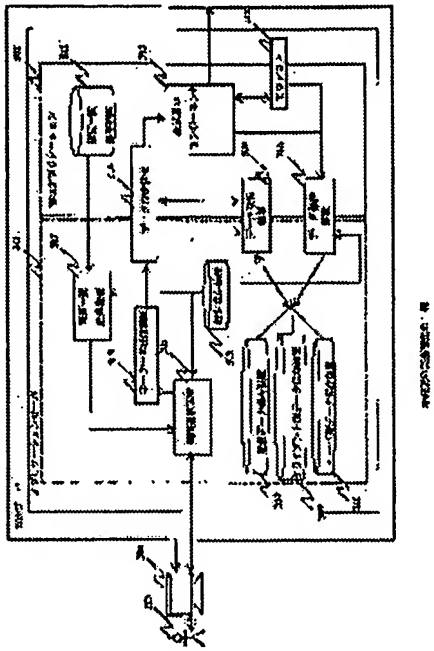
(51)Int.Cl. G06F 11/34

(21)Application number : 2001-371432
(22)Date of filing : 05.12.2001

(71)Applicant : HITACHI SOFTWARE ENG CO LTD
(72)Inventor : TSUKURIDA TETSUO

(54) APPARATUS AND METHOD FOR SUPPORTING DEVELOPMENT OF WEB APPLICATION

(57)Abstract:
PROBLEM TO BE SOLVED: To reduce the amount of operation required of a developer regarding screen transition and man hours for data inputs by adding a screen to be developed or tested into the process of screen transition, and, even if the screen uses data retained by a server, creating data for which a Web application is retained by the server.
SOLUTION: A client data retention device 109, a transmit data retention device 308, and a server data retention device 310 retain data in such a way that the data required and keywords are defined to match. The applications developer 303 obtains a URL which the Web application to be developed discloses and its keyword retained in either of the retention devices, transmits a keyword request to the server along with the URL, obtains the data corresponding to the keyword from either of the data retention devices, and sets the obtained data in a domain used by the Web application, prior to processing of the Web application.



LEGAL STATUS

[Date of request for examination] 05.07.2004
[Date of sending the examiner's decision of rejection] 24.07.2007
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号
特開2003－173275
（P2003－173275A）
(43)公開日　平成15年６月20日(2003.6.20)

(51)Int.Cl.⁷	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　11/34		Ｇ０６Ｆ　11/34	Ａ　５Ｂ０４２

審査請求　未請求　請求項の数８　ＯＬ　（全 27 頁）

(21)出願番号　特願2001－371432(P2001－371432)

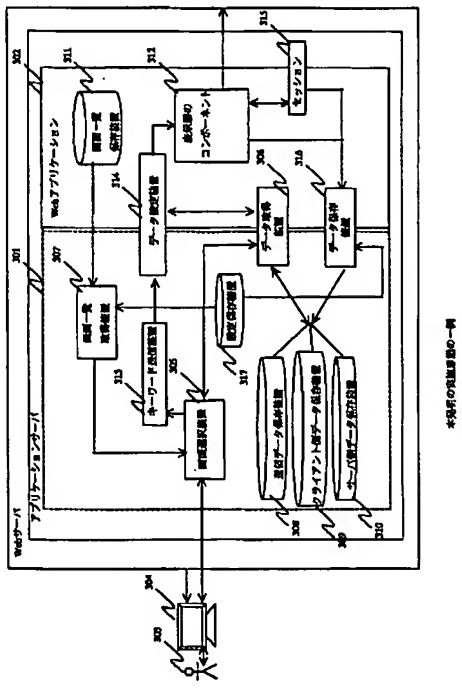
(22)出願日　平成13年12月５日(2001.12.5)

(71)出願人　000233055
日立ソフトウエアエンジニアリング株式会社
神奈川県横浜市鶴見区末広町一丁目１番43
(72)発明者　造田　哲雄
神奈川県横浜市中区尾上町６丁目81番地
日立ソフトウエアエンジニアリング株式会社内
(74)代理人　100096954
弁理士　矢島　保夫

Ｆターム(参考)　5B042 GA12 GA18 GB02 GC10 GC12
HH01 HH11 HH30 MC09 MC36
NN11 NN29

(54)【発明の名称】　Ｗｅｂアプリケーション開発支援装置及び方法

(57)【要約】
【課題】画面遷移の途中に開発やテストの対象となっている画面が含まれており、その画面がサーバ側に保持されているデータを使用する画面であっても、Webアプリケーションがサーバ側に保持されているデータを生成することにより、開発者にかかる画面遷移に伴う操作及びデータ入力の工数を削減することができるようにする。
【解決手段】クライアント側データ保存装置309、送信データ保存装置308、及びサーバ側データ保存装置310で、必要なデータとキーワードとを対応付けて定義して保存する。アプリケーション開発者303は、開発対象のWebアプリケーションが公開しているＵＲＬ及び前記保存装置で保存されているキーワードを取得し、ＵＲＬと共にキーワードの要求をサーバに送信し、キーワードに対応するデータを前記いずれかのデータ保存装置から取得し、該当Webアプリケーションの処理前に、該当Webアプリケーションが使用する領域に取得したデータを設定する。



【特許請求の範囲】
【請求項１】クライアントとサーバ間でデータのやり取りを行うWebアプリケーションの開発を支援するWebアプリケーション開発支援装置であって、
ブラウザが送信するデータとキーワードとを対応付けて定義し、クライアント側データとして保存するクライアント側データ保存手段を備えることを特徴とするWebアプリケーション開発支援装置。
【請求項２】クライアントとサーバ間でデータのやり取りを行うWebアプリケーションの開発を支援するWebアプリケーション開発支援装置であって、
Webアプリケーションの処理後にクライアントに送信するデータとキーワードとを対応付けて定義し、送信データとして保存する送信データ保存手段を備えることを特徴とするWebアプリケーション開発支援装置。
【請求項３】クライアントとサーバ間でデータのやり取りを行うWebアプリケーションの開発を支援するWebアプリケーション開発支援装置であって、
サーバ側で保存するデータとキーワードとを対応付けて定義し、サーバ側データとして保存するサーバ側データ保存手段を備えることを特徴とするWebアプリケーション開発支援装置。
【請求項４】請求項１から３の何れか１つに記載のWebアプリケーション開発支援装置において、
開発対象である前記Webアプリケーションが処理を行うために必要なデータを、前記クライアント側データ保存手段、前記送信データ保存手段、および／またはサーバ側データ保存手段から取得し、前記Webアプリケーションの処理前に、そのWebアプリケーションが使用する領域に設定する手段をさらに備えることを特徴とするWebアプリケーション開発支援装置。
【請求項５】請求項４に記載のWebアプリケーション開発支援装置において、
開発対象である前記Webアプリケーションが公開しているＵＲＬの一覧を取得する手段と、
前記クライアント側データ保存手段、前記送信データ保存手段、および／またはサーバ側データ保存手段に保存されているキーワードの一覧を取得する手段と、
取得したＵＲＬ一覧から起動するＵＲＬをユーザに選択させる手段と、
取得したキーワード一覧から、前記起動するＵＲＬで使用するデータに対応するキーワードをユーザに選択させる手段と、
選択されたＵＲＬおよび選択されたキーワードに基づいて、そのキーワードに対応するデータを取得し、前記Webアプリケーションが使用する領域に設定する手段とをさらに備えることを特徴とするWebアプリケーション開発支援装置。
【請求項６】クライアントとサーバ間でデータのやり取りを行うWebアプリケーションの開発を支援するWebアプリケーション開発支援方法であって、
ブラウザが送信するデータとキーワードとを対応付けて定義し、クライアント側データとして保存するクライアント側データ保存ステップと、
Webアプリケーションの処理後にクライアントに送信するデータとキーワードとを対応付けて定義し、送信データとして保存する送信データ保存ステップと、
サーバ側で保存するデータとキーワードとを対応付けて定義し、サーバ側データとして保存するサーバ側データ保存ステップとを備えることを特徴とするWebアプリケーション開発支援方法。
【請求項７】請求項６に記載のWebアプリケーション開発支援方法において、
開発対象である前記Webアプリケーションが処理を行うために必要なデータを、前記クライアント側データ保存ステップ、前記送信データ保存ステップ、および／またはサーバ側データ保存ステップで保存したデータの中から取得し、前記Webアプリケーションの処理前に、そのWebアプリケーションが使用する領域に設定するステップをさらに備えることを特徴とするWebアプリケーション開発支援方法。
【請求項８】請求項７に記載のWebアプリケーション開発支援装置において、
開発対象である前記Webアプリケーションが公開しているＵＲＬの一覧を取得するステップと、
前記クライアント側データ保存ステップ、前記送信データ保存ステップ、および／またはサーバ側データ保存ステップで保存されたキーワードの一覧を取得するステップと、
取得したＵＲＬ一覧から起動するＵＲＬをユーザに選択させるステップと、
取得したキーワード一覧から、前記起動するＵＲＬで使用するデータに対応するキーワードをユーザに選択させるステップと、
選択されたＵＲＬおよび選択されたキーワードに基づいて、そのキーワードに対応するデータを取得し、前記Webアプリケーションが使用する領域に設定するステップとをさらに備えることを特徴とするWebアプリケーション開発支援方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、クライアントから送信されたデータ及びサーバに保持されているデータを使用して処理を行い、画面遷移を行うWebアプリケーションの開発を支援するWebアプリケーション開発支援装置及び方法に関する。
【０００２】
【従来の技術】インターネットやイントラネット上のWebアプリケーションには、乗車券予約や検索、本や日用品の販売のようなサービスを提供するものがある。通

常、これらのサービスは、認証画面→予約画面→確認画面のように画面遷移を行って処理を進めている。通常のWebアプリケーションでは、画面遷移に伴ってブラウザに入力されたデータを遷移した先の画面で参照したり、加工したり、表示したりしている。また、インターネットやイントラネットで使用されている通信方式では、画面間でデータを保持する機能を持たない。しかし、複数画面でデータの共有を実現するために、ブラウザの要求を受け取るサーバには、後の処理に必要になるデータなどを保持しておく機能がある。

【０００３】従来、このようなWebアプリケーションを開発する場合、画面遷移の順番に開発を行っている。従来技術を使用した開発例として、例えば特開平10-275093号公報に記載のような、各画面でのデータ入力を支援するためにアプリケーションへの操作を記録しておく方法がある。この操作記録から、ユーザがブラウザに入力したデータと入力した位置を取り出し、その情報を用いてデータ入力を自動化する。またデータ入力により、アプリケーションが出力したデータを使ってテストを行ったり、以前の処理結果と比較したりすることができる。

【０００４】

【発明が解決しようとする課題】しかしながら、前述特開平10-275093号公報に記載の技術は、ユーザが入力したデータと入力した位置など、クライアント側のアプリケーションが出力する情報は保存できるが、サーバにはアクセスしないため、Webアプリケーションのようにプログラムがサーバで実行しているアプリケーションでサーバに保存されるデータを保存することができないという問題があった。そのため、該当Webアプリケーションが、サーバに保存されているデータを使用する場合、サーバに保存されているデータを生成するために、画面遷移に沿って目的画面へ到達しなければならず、開発が終了するまで同じ操作や同じデータ入力を何度も繰り返さなければならないという問題がある。

【０００５】本発明は、上記問題点に鑑み、画面遷移の途中に開発やテストの対象となっている画面が含まれており、その画面がサーバ側に保持されているデータを使用する画面であっても、Webアプリケーションがサーバ側に保持されているデータを生成することにより、開発者にかかる画面遷移に伴う操作及びデータ入力の工数を削減することを目的とする。

【０００６】

【課題を解決するための手段】上記目的を達成するため、請求項１に係る発明は、クライアントとサーバ間でデータのやり取りを行うWebアプリケーションの開発を支援するWebアプリケーション開発支援装置であって、ブラウザが送信するデータとキーワードとを対応付けて定義し、クライアント側データとして保存するクライアント側データ保存手段を備えることを特徴とする。

【０００７】請求項２に係る発明は、クライアントとサーバ間でデータのやり取りを行うWebアプリケーションの開発を支援するWebアプリケーション開発支援装置であって、Webアプリケーションの処理後にクライアントに送信するデータとキーワードとを対応付けて定義し、送信データとして保存する送信データ保存手段を備えることを特徴とする。

【０００８】請求項３に係る発明は、クライアントとサーバ間でデータのやり取りを行うWebアプリケーションの開発を支援するWebアプリケーション開発支援装置であって、サーバ側で保存するデータとキーワードとを対応付けて定義し、サーバ側データとして保存するサーバ側データ保存手段を備えることを特徴とする。

【０００９】請求項４に係る発明は、請求項１から３の何れか１つに記載のWebアプリケーション開発支援装置において、開発対象である前記Webアプリケーションが処理を行うために必要なデータを、前記クライアント側データ保存手段、前記送信データ保存手段、および／またはサーバ側データ保存手段から取得し、前記Webアプリケーションの処理前に、そのWebアプリケーションが使用する領域に設定する手段をさらに備えることを特徴とする。

【００１０】請求項５に係る発明は、請求項４に記載のWebアプリケーション開発支援装置において、開発対象である前記WebアプリケーションがＵＲＬの一覧を取得する手段と、前記クライアント側データ保存手段、前記送信データ保存手段、および／またはサーバ側データ保存手段に保存されているキーワードの一覧を取得する手段と、取得したＵＲＬ一覧から起動するＵＲＬをユーザに選択させる手段と、取得したキーワード一覧から、前記起動するＵＲＬで使用するデータに対応するキーワードをユーザに選択させる手段と、選択されたＵＲＬおよび選択されたキーワードに基づいて、そのキーワードに対応するデータを取得し、前記Webアプリケーションが使用する領域に設定する手段とをさらに備えることを特徴とする。

【００１１】請求項６に係る発明は、クライアントとサーバ間でデータのやり取りを行うWebアプリケーションの開発を支援するWebアプリケーション開発支援方法であって、ブラウザが送信するデータとキーワードとを対応付けて定義し、クライアント側データとして保存するクライアント側データ保存ステップと、Webアプリケーションの処理後にクライアントに送信するデータとキーワードとを対応付けて定義し、送信データとして保存する送信データ保存ステップと、サーバ側で保存するデータとキーワードとを対応付けて定義し、サーバ側データとして保存するサーバ側データ保存ステップとを備えることを特徴とする。

【００１２】請求項７に係る発明は、請求項６に記載のWebアプリケーション開発支援方法において、開発対象である前記Webアプリケーションが処理を行うために必

要なデータを、前記クライアント側データ保存ステップ、前記送信データ保存ステップ、および／またはサーバ側データ保存ステップで保存したデータの中から取得し、前記Webアプリケーションの処理前に、そのWebアプリケーションが使用する領域に設定するステップをさらに備えることを特徴とする。

【００１３】請求項８に係る発明は、請求項７に記載のWebアプリケーション開発支援装置において、開発対象である前記Webアプリケーションが公開しているＵＲＬの一覧を取得するステップと、前記クライアント側データ保存ステップ、前記送信データ保存ステップ、および／またはサーバ側データ保存ステップで保存されたキーワードの一覧を取得するステップと、取得したＵＲＬ一覧から起動するＵＲＬをユーザに選択させるステップと、取得したキーワード一覧から、前記起動するＵＲＬで使用するデータに対応するキーワードをユーザに選択させるステップと、選択されたＵＲＬおよび選択されたキーワードに基づいて、そのキーワードに対応するデータを取得し、前記Webアプリケーションが使用する領域に設定するステップとをさらに備えることを特徴とする。

【００１４】

【発明の実施の形態】以下、図面を参照して本発明の実施の形態を具体的に説明する。

【００１５】図１は、開発対象とするWebアプリケーションの構成の一例である。以下、図１に示したWebアプリケーション構成において、ユーザ101がブラウザ102を操作してから、処理結果を表示するまでの流れを説明する。

【００１６】ユーザ101は、端末を使ってブラウザ102を表示し、そのブラウザ102から送信されたＵＲＬへアクセスする。そのとき、ユーザ101によって入力されたデータやブラウザ102の情報など、クライアント側にある情報も要求として、サーバ103に用意されているWebサーバ104に送信する。ブラウザ102からの要求を受信したWebサーバ104は、Webアプリケーション110を実行する環境が整っているアプリケーションサーバ105を使用して、Webアプリケーション110を実行するための窓口となる表示層のコンポーネント106を実行する。アプリケーションサーバ105により実行された表示層のコンポーネント106は、ブラウザ102から送信されたデータや、アプリケーションサーバ105が、Webアプリケーションが使用するデータをサーバ103に保持する機能を持つセッション107から、情報を取得して、業務処理を実行する。ここで、業務処理とは、例えばユーザを認証する処理や、データベース108にアクセスしてデータを取得する処理や、すでに実装されたJavaのクラスなどの実行可能なコンポーネント109を呼び出したりして、開発対象のWebアプリケーションに実装された顧客の要求を満たす一連の処理のことを指す。サーバ103にデータを保存する機能を持つセッション107は、Webアプリケーション110の仕様により、ブラウザ102から送信されたデータや業務処理により得られた結果を、例えば、次の画面でも使用するし、そのまた次の画面でも使用するというように、画面遷移に沿って引き続き使用する必要がある場合に多く利用される。

【００１７】表示層のコンポーネント106は、要求された業務処理を実行したあと、処理結果をブラウザ102に表示するために、HTML形式のデータを生成し、Webサーバ104経由でブラウザ102に送信する。このHTML形式のデータは、表示層のコンポーネント106により処理された結果を含んでいる。例えば、ユーザIDを使用してデータベース108から取得したユーザ名や、実行可能なコンポーネント109で行った認証結果、検索条件に合った検索結果の一覧などである。本実施形態では、それらの情報の保持には、ブラウザ102から送られてくるデータやセッション107のデータを取得する場合と同じ方法を使うようにするため、動的なデータをキーワードとそのキーワードに対応するデータというように対にして保持することとする。ブラウザ102は、受信したHTML形式のデータをブラウザ102に表示する。以上でユーザ101が処理を要求し、処理結果をブラウザ102に表示するまでの一連の処理が終了する。

【００１８】図２は、キーワードとデータとデータの種類の対応表の一例を示す図である。同図において、キーワード201は、データファイルに保存するときに、値と対応付けるための文字列を記述する。変数202は、データファイルに保存するときにデータが格納されている変数名を記述する。データの種類203は、格納するデータファイルの種類を記述する。例えば、受信して、かつセッション107にも保存するようなデータの場合は、複数のデータファイルの種類を指定しても良い。図２の場合、ユーザ名を格納する変数userName204のデータは、受信し、かつセッションデータに保存するので、データの種類203としてセッション及び受信205を指定している。ユーザ名を取得したり、格納したりするときのキーワードはUSERNAME 206である。

【００１９】図３は、本発明のWebアプリケーション開発支援装置を適用したシステムの一例を示す図である。点線301の中に示しているものが、本発明に関わる部分（以下、開発支援部301と呼ぶ）であり、実線302の中に示しているものが開発対象のWebアプリケーションである。本実施形態では、本発明をWebアプリケーション301として具現化し、該当Webアプリケーション302と同じアプリケーションサーバ上に構成した。これは、Webアプリケーションが異なってもWebサーバが同じであれば、各Webアプリケーションが同一のセッションを使用することが可能なためである。

【００２０】以下、本実施形態の動作概要を説明しながら、各部分の機能について説明する。

【００２１】まず、アプリケーション開発者303が開発を行うために使用する端末304を用いて、本実施形態のシステムを起動する。端末304には、本発明に係る開発支援部や、OSや、開発に必要なアプリケーションが動作するために必要なメモリ、及び外部記憶装置などが付随している。アプリケーション開発者303によって起動された開発支援部301は、まず画面選択装置305を起動する。

【００２２】起動された画面選択装置305は、データ取得装置306を使用して、Webアプリケーション302の表示層のコンポーネント312がブラウザに送信するデータを保存している送信データ保存装置308や、Webアプリケーション使用者が入力したデータを保存しているクライアント側データ保存装置309や、Webアプリケーションが使用するためのデータをサーバ側に保存しているサーバ側データ保存装置310などから、それぞれデータを取得するためのキーワードを取得する。また、画面選択装置305は、画面一覧取得装置307を使用して、画面一覧保存装置311から該当Webアプリケーション302が公開している画面にアクセスするＵＲＬの一覧を取得する。画面選択装置305は、データ取得装置306と画面一覧取得装置307を使って取得した情報を端末304に表示する。

【００２３】アプリケーション開発者303は、端末304に表示されたWebアプリケーション302の画面へアクセスするためのＵＲＬ一覧や、Webアプリケーション302の表示層のコンポーネント312の処理結果のデータや、Webアプリケーション使用者が入力したデータや、Webアプリケーションが処理のためにサーバに保存するデータなどを選択する。

【００２４】画面選択装置305は、アプリケーション開発者303により選択されたWebアプリケーションにアクセスする。そのとき、画面選択装置305は、キーワード送信装置313を起動して、アプリケーション開発者303が画面選択装置305で選択したＵＲＬで使用する各データを取得するためのキーワードを、キーワード送信装置313を使って、データ設定装置314に送信する。

【００２５】データ設定装置314は、受信したキーワードを使って、送信データ保存装置308や、クライアント側データ保存装置309や、サーバ側データ保存装置310などからデータを取得する。そして、アプリケーション開発者303により選択されたWebアプリケーションがサーバ側でデータを利用するために、取得したサーバ側データを、該当Webアプリケーションが業務処理を始める前にセッション315に設定する。この機能により、画面遷移に沿って処理を行ってきたときと同様のデータをセッション315に保持することができる。その後、データ設定装置314は、クライアント側データおよび送信データを、Webアプリケーション302が使用する領域に、上書きや置き換えや設定する。このあとの業務処理では、これらの機能により、表示層のコンポーネント312は、画面遷移に沿って処理してきた結果のデータを使って業務処理を行うことができる。従って、これまでに設定されたデータを使用して業務処理を行うように、Webアプリケーション開発者303は、表示層のコンポーネント312をWebアプリケーションの仕様に従って実装すればよい。

【００２６】表示層のコンポーネント312の業務処理が終了したら、表示層のコンポーネント312が受信したデータをクライアント側データ保存装置309に、表示層のコンポーネント312がセッション315に保存するデータをサーバ側データ保存装置310に、表示層のコンポーネント312がブラウザに送信するデータを送信データ保存装置308に、それぞれ保存するために、データ保存装置316を起動する。データ保存装置316は、設定保存装置317から、データを保存するファイル名称の定義を取得し、それを使って、そのデータにキーワードをつけてファイル名を作成して各保存装置に保存する。

【００２７】その後、表示層のコンポーネント312の処理結果をHTML形式のデータとして端末304に送信することで、アプリケーション開発者303が選択したＵＲＬで、Webアプリケーション302が処理を行った結果の画面が表示される。また、例えば、セッション315へのデータのように表示層のコンポーネント312が処理を行った結果をサーバ側データ保存装置310に保存するので、画面選択装置305で選択すれば、そのデータを次の画面に対応しているＵＲＬに送るデータとして利用することができる。

【００２８】また、オブジェクト指向では、特性をまとめたクラスという概念があり、スーパークラスを継承して作成されるサブクラスは、スーパークラスで定義された機能を引き継いで使用することができる。そのため、サーブ レットでの処理順序を決定するための処理や、JavaBeansを呼び出す処理を、オブジェクト指向の技術であるスーパークラスであらかじめ定義しておき、実際の開発では、そのスーパークラスを継承したサブクラスの開発を行う。また、サーブレットが呼び出すJavaBeansは、Webアプリケーションの設計工程で導出され、サーブレットは必要に応じてJavaBeansにアクセスして処理を進める。本実施形態では、データ設定装置314、データ取得装置306、データ保存装置316をサーブレットのスーパークラスに操作として実装して提供し、開発にはスーパークラスを継承したサブクラスで処理を記述する。それを表すため、これらの装置は点線301と実線302に跨って記述している。

【００２９】次に、本実施形態における処理を、図４に示す該当Webアプリケーションのサーブレットを起動するためのＵＲＬと、そのサーブレットが使用するデータをキーワードと共に格納しているデータファイルを選択して、本発明に係る開発支援部301に要求を出すまでの処理と、図１４に示す要求を受け取った開発支援部301がWebアプリケーションが使用する領域にデータを設定

し、サーブレットが業務処理を行って、開発支援部301がデータファイルを生成するという一連の処理が終了するまでの２つの処理に分けて説明する。

【００３０】まず、図４について説明する。図４は、本実施形態によりWebアプリケーション開発を行う場合の処理の一部を表すフローチャートである。ステップ401，402及び403は、アプリケーション開発者が行うWebアプリケーション開発では一般的なステップである。このWebアプリケーションの開発では、Javaを用いたWebアプリケーションで表示層のコンポーネントに使用できる技術の中で、データベースにアクセスする部分や既存のコンポーネントにはJavaBeansという技術を使い、JavaBeansの処理結果を加工したりブラウザに表示する画面のレイアウトを決定するコンポーネントにはサーブレットという技術を用いる。まず、図４に記述しているステップ401からステップ405の処理を説明したあと、図１１及び１２に示す図４の処理で表示された画面を使って、ステップ406からステップ409を説明する。

【００３１】まず、ステップ401で、開発対象とする画面遷移を決定する。画面遷移は、Webアプリケーションの設計工程で作成される。

【００３２】図５は、Webアプリケーションで発生する画面遷移の一例である。角丸四角形（例えば501）は、Webアプリケーションが表示する画面を表す記号である。また、角丸四角形内に記述している文字列（例えば502）は、Webアプリケーションでの画面名を表している。そして、矢印（例えば503）は、画面遷移を表している。図５に示した例の場合、該当Webアプリケーションは、「開始」という画面名の画面から、「登録処理」という画面名の画面、または、「商品選択」という画面名の画面に遷移することを表している。本実施形態では、開発対象とする画面遷移として、「開始」→「商品選択」→「商品確認」→「終了」という画面遷移を選択する。

【００３３】ステップ401でアプリケーション開発者が開発対象とする画面遷移を選択したら、ステップ402で開発対象画面を選択する。通常、画面遷移を持つWebアプリケーション開発では、画面遷移の先頭から開発を行う。本実施形態の場合では、「開始」を最初の実装対象として選択する。

【００３４】ステップ402で開発対象画面を選択したら、ステップ403で画面を開発する。このステップでは、該当Webアプリケーションの設計工程で作成された該当Webアプリケーションの仕様に基づきプログラムを実装する。このステップでは、通常のテキストエディタを使ったり、他の開発支援システムを用いてもよく、作成したテキスト形式のプログラムがエラーなくコンピュータが実行することができるバイトコードになればよい。

【００３５】ここで、図６を参照して、サーブレットを開発するために必要なWebアプリケーションの設計工程で導出される画面名とサーブレットとＵＲＬとの対応表について説明する。Webアプリケーション開発者は、画面名とサーブレットとＵＲＬの対応表から、開発するサーブレットのクラス名とＵＲＬを参照する。画面名601は、アプリケーション開発者がWebアプリケーションの設計工程で定義した画面名を記述する。サーブレット602は、画面名601の画面を表示するために実行するサーブレットのクラス名を定義する。ＵＲＬ603は、サーブレット602にブラウザがアクセスするためのアプリケーションサーバに依存しない部分のＵＲＬを定義する。図６の場合、ブラウザが画面名称「開始」604を表示するための、アプリケーションサーバに依存しない部分のＵＲＬは「/Index」605であり、その画面を表示するために作成しなければならないサーブレットは「IndexServlet」606であることを表している。本実施形態の場合、開発対象サーブレットが「IndexServlet」606であれば、その「IndexServlet」を使い慣れたエディタなどを用いてプログラミングし実装する。

【００３６】ここで本実施形態で提供するスーパークラスと、そのスーパークラスを継承して該当Webアプリケーションのサーブレットを作成する場合の例について説明する。

【００３７】図７は、サーブレットのスーパークラスの実装の一例である。クラス名BaseServlet701は、本実施形態において提供されているサーブレットのスーパークラスの名称である。doGet()702は、ブラウザに要求があったとき、サーブレットがその要求を受け付ける窓口になる操作のひとつである。操作doGet()702には、クライアント側データの値を取得し定義した変数に代入する処理Hashtable clientData = getClientData(req);703と、送信データを取得し定義した変数に代入する処理Hashtable sendData = getSendData(req);704と、サーバ側データを取得し定義した変数に代入する処理Hashtable sessionData = getSessionData(req);705を記述している。

【００３８】操作getClientData();706は、クラスBaseServlet701内で定義される操作のひとつで、クライアントから送られてきた要求を取得し、Javaが提供しているクラスのひとつであるHashtableクラスで返却する処理が実装されている。操作getSendData();707は、クラスBaseServlet701内で定義される操作のひとつで、送信データとして格納されているデータを取得し、Javaが提供しているクラスのひとつであるHashtableクラスで返却する処理が実装されている。操作getSessionData();708は、クラスBaseServlet701内で定義される操作のひとつで、サーバ側データに格納されているデータを取得し、Javaが提供しているクラスのひとつであるHashtableクラスで返却する処理が実装されている。これら3つの操作が行う、データを取得して該当Webアプリケーション

が使用する領域に設定する処理については、後で説明する。

【００３９】本実施形態では、該当Webアプリケーションの各画面に対応したサーブレットをBaseServlet701を継承したサブクラスとして実装する操作doBusiness()709に、Webアプリケーションの設計工程で導出された業務処理を記述することで、サブクラスで記述された業務処理が行われる。操作doBusiness()709は、ブラウザに返すHTML形式のデータをJavaが提供しているクラスのひとつのStringクラスで返却し、定義した変数htmlStreamに代入する処理を記述している(行710)。

【００４０】操作setClientData();711は、BaseServlet701で定義する処理のひとつで、クライアントから送られてきた要求をキーワードと対応付けてクライアント側データ保存装置309に格納する処理を行う。操作setSendData();712は、BaseServlet701で定義する処理のひとつで、doBusiness()709の処理結果を、送信データ保存装置308に保存する処理を行う。操作setSessionData();713は、BaseServlet701で定義する処理のひとつで、操作doBusiness()709の処理結果をサーバ側データ保存装置310に保存する処理を行う。本実施形態の場合、各操作の引数に与えた、Hashtableクラスのオブジェクトをそのままシリアライズして、各データ保存装置に保存する。データを保存するこれらの操作の処理については後で説明する。

【００４１】最後に、該当サーブレットの業務処理が終了したHTML形式のデータをブラウザに送信する処理(行714)を記述している。

【００４２】上記処理をスーパークラスであるBaseServlet701に記述することにより、BaseServlet701を継承したサブクラスは、BaseServlet701に記載された処理順序で処理を行う。

【００４３】次に、BaseServlet701を継承して作成するサブクラスの実装について説明する。図８は、サーブレットのサブクラスの実装の一例である。

【００４４】行public class IndexServlet extends BaseServlet 801の記述では、本実施形態で提供しているサーブレットのスーパークラスBaseServletを継承して、開始画面のサーブレットのクラスIndexServlet 802を定義している。

【００４５】doBusiness();803には、すでに説明したように、開始画面が行う処理を記述している。本実施形態の場合、開始画面では、ユーザが入力したユーザIDを受け取り、そのユーザIDを使ってデータベースからユーザ名を取得し、ブラウザに表示するという業務処理を記述している。

【００４６】String userID = (String)clientData.get("USERID");804は、ブラウザが送信したデータを操作doBusiness();803の引数で渡されるクライアント用データの変数「clientData」から取得して、変数useIDに代入する処理である。また、String userName = getUserName(userID);805は、変数userIDを使って、データベースからユーザ名を取得して、userNameに代入する処理である。操作getUserName()は、クラスIndexServlet802内で定義され、その処理中には、JavaBeansを使って、データベースにアクセスして、ユーザ名を取得する処理が記述されている。

【００４７】sendData.put("USRENAME", userName);806は、取得したユーザ名を送信データ保存装置308に保存するために、送信データ用の変数「sendData」にキーワード「USERNAME」で格納する処理である。また、sessionData.put("USRENAME", userName);807は、行805で取得したユーザ名を、サーバ側データとして、セッションに保存するために、キーワード「USERNAME」を対応付けて格納する処理である。これら、業務処理に対応付けるキーワードは、Webアプリケーションの設計工程で、アプリケーション開発者が定義するものである。

【００４８】行808は、ブラウザにユーザ名を表示するためのHTML形式のデータを生成する処理である。本実施形態の場合、動的に変更して表示するデータのユーザ名を、行805の業務処理で取得しているので、ユーザ名の変数userName809をHTML形式のデータに埋め込む処理を記述している。そして、行return htmlStream; 810で、生成したHTML形式のデータを返却している。

【００４９】再び図４を参照して、Webアプリケーション開発の手順の説明を続ける。ステップ404では実装が完了したか判別し、未だ完了していないときはステップ401に戻る。これは、ステップ405に進む前にステップ401からステップ403の処理を繰り返し行い、Webアプリケーションでアクセスできるサーブレットのクラスを複数個作成しておいてもよいことを表している。本実施形態の場合、図７、８に示されているサーブレットは実行できるようになっているとする。

【００５０】ステップ405では、ステップ401からステップ404の処理でコンピュータで実行できる形式になった１つ以上のサーブレットを、Webアプリケーションを実行する環境が用意されているアプリケーションサーバに配置する処理を行う。配置とは、Webアプリケーションをアプリケーションサーバが実行できるように準備することである。本実施形態で用いるJ2EEのサーブレットAPIの仕様では、Webアプリケーションは、Webアプリケーションアーカイブという拡張子が「.war」の特定のフォルダ構成と設定ファイルを持ったファイルを作成し、アプリケーションサーバに固有のやり方でアプリケーションサーバに配置される。本実施形態の場合、本発明に係るWebアプリケーション開発支援装置を具現化しているのは、サーブレットに処理を定義しているスーパークラスBaseServletを提供すると共に、該当WebアプリケーションにWebサーバおよびアプリケーションサーバを通じてアクセスするWebアプリケーション（開発支援部301）

である。従って、本発明に係るWebアプリケーション開発支援装置も、開発時に使用するアプリケーションサーバに配置し使用することができる。

【００５１】次に、該当Webアプリケーションが公開しているサーブレットとインターネットやイントラネットからアクセスするためのＵＲＬとの対応付けを定義する方法を説明する。この対応付けは、J2EEのサーブレットAPIの仕様では、「web.xml」というファイルに定義する。図９は、web.xmlの内容の一例である。web.xmlはXMLの文法に従って記述する。XMLの文法では、開始タグと終了タグが存在し、タグ名を「タグ名」としたとき、開始タグ、終了タグの各タグの表記法は、それぞれ「<;タグ名>;」、「<;/タグ名>;」である。各タグを使って定義する文字列は、開始タグと終了タグの間に記述する。以降、特に断らない限り、開始タグと用いてタグの説明を行う。

【００５２】まず、<;servlet>;タグ901により、ファイルweb.xml内でサーブレットを識別するための識別子を<;servlet-name>;タグ902で定義し、<;servlet-name>;タグ902で定義した識別子に対応するサーブレットのクラスを<;servlet-class>;タグ903で定義する。本実施形態の図９の場合、ファイルweb.xml内でサーブレットを識別するための識別子indexServlet 904に対応するサーブレットのクラスは、IndexServlet 905である。

【００５３】<;servlet-mapping>;タグ906で、<;servlet-name>;タグ902によって定義されたファイルweb.xml内でサーブレットを識別するための識別子と、<;url-pattern>;タグ907で定義されたサーブレットにアクセスするためのＵＲＬとを、対応付ける。本実施形態の場合、ファイルweb.xml内でサーブレットを識別するための識別子indexServlet 904に対するサーブレットにアクセスするためのＵＲＬを、図６の対応表を参照して、/Index 908と定義している。

【００５４】本実施形態で挙げた図９では、簡単のため、説明に必要十分な最低部分として、「開始」画面に対する定義のみ示しているが、実際には、該当Webアプリケーションが公開しているＵＲＬ全てを定義する。具体的には、<;servlet>;901タグから終了タグ<;/servlet>;までのまとまり909を必要な回数繰り返して、該当Webアプリケーションが表示する全ての画面に対し定義した後で、<;servlet-mapping>;タグ906から終了タグ<;/servlet-mapping>;までのまとまり910を必要な回数繰り返して定義する。なお「http://java.sun.com/j2ee/dtds/web-app_2_2.dtd」911は、web.xmlを記述する場合に必要となる、規約で決めらている文字列である。

【００５５】再び図４を参照して、ステップ405で該当Webアプリケーションの配置が終了したら、ステップ406で、該当Webアプリケーションを指定して本発明に係る開発支援部301を起動する。本実施形態のように本発明をWebアプリケーションという形式で具現化した場合、本発明を起動するということは、ブラウザから本発明に係る開発支援部301へＵＲＬを指定してアクセスするということである。また、ひとつのアプリケーションサーバには複数のWebアプリケーションを配置することができるので、該当Webアプリケーションの指定は、開発対象のWebアプリケーションを、サーブレットの引数として指定したり、設定ファイルなどを使って設定したりすればよい。本実施形態の場合は、サーブレットの引数として、該当Webアプリケーションを指定する。

【００５６】本実施形態の場合、該当WebアプリケーションをWebアプリケーションアーカイブ形式で配置し、該当WebアプリケーションのWebアプリケーションアーカイブのファイル名が「sample.war」であり、そのファイルが置かれているフォルダが「c:¥project」であるとし、さらに本発明に係る開発支援部301が該当Webアプリケーションアーカイブファイルを取得するためのキーワードが「WAR」であるとした場合は、Webアプリケーションアーカイブを指定する文字列「c:¥project¥sample.war」を、ＵＲＬに使うことができない文字をＵＲＬで使うことができるようにする変換方式である「x-www-form-ＵＲＬ符号化形式」に従って変換し、本発明に係る開発支援部301を起動するサーブレットのＵＲＬに引数として指定して付け加えて、例えば「http://localhost:8080/Index?WAR=c%3A%5Cproject%5Csample.war」1102としWebアプリケーションを特定して起動する。

【００５７】ステップ406により起動された本発明に係る開発支援部301は、画面選択装置305に、該当Webアプリケーションが公開している画面と、ユーザが入力したデータをクライアント側データ保存装置309から取得するためのキーワードとなるデータファイルの名称と、サーバ側データ保存装置310から取得するためのキーワードである保存されているデータファイルの名称と、サーバからクライアントに対して処理結果として送信されるデータを送信データ保存装置308から取得するためのキーワードとなるデータファイルの名称との一覧を表示する。そして、ステップ407で、アプリケーション開発者303が、該当WebアプリケーションのサーブレットにアクセスするためのＵＲＬ及びそのサーブレットが業務処理で使用するデータを選択する。

【００５８】図１０を用いて、ステップ406の処理の詳細を説明する。

【００５９】まず、ステップ1001で、指定された該当Webアプリケーションアーカイブファイルに格納されているファイルの中からweb.xmlを取得する。J2EEのサーブレットAPIの仕様により、web.xmlはWebアプリケーションアーカイブファイルを展開したら作成されるWEB-INFフォルダのすぐ下に格納されているので、web.xmlを見つけることは容易にできる。また、本実施形態のように、開発支援部301をJavaで開発したWebアプリケーションとして提供する場合、Javaでは、Webアプリケーショ

ンアーカイブ形式のファイルを展開したり、格納されているファイルを取得できるクラスや操作が用意されているので、web.xmlの取得は容易にできる。

【００６０】次にステップ1002で、ファイルweb.xmlを解析して、ブラウザからWebアプリケーションにアクセスするためのＵＲＬを取得する。J2EEのサーブレットAPIの仕様によりweb.xmlはＸＭＬ形式で記述されることが決められているので、すでに説明したように、タグ<;url-pattern>;で定義されている文字列を取得することで、上記機能は容易に実現することができる。例えば、本実施形態のように、開発支援部301をJavaで開発したWebアプリケーションとして提供する場合、Javaでは、ＸＭＬファイルの解析を容易にできるようにクラスや操作が提供されているので、それを用いることで、ＵＲＬを簡単に取得することができる。また、アプリケーションサーバの設定によっては、web.xmlから取得したＵＲＬだけではサーブレットにアクセスできないことがあるので、画面一覧取得装置307は、必要な文字列を追加してサーブレットにアクセスできるようにする機能を持っている。これらアプリケーションサーバに固有な設定は、該当Webアプリケーションの設計工程で分かるので、設定ファイルやオプション、使用するスーパークラスを変えたりすることであらかじめ指定しておく。

【００６１】例えば、本実施形態の場合で、tomcatといわれるWebサーバとアプリケーションサーバが一体となっているアプリケーションを採用したとすると、tomcatではデフォルトで、web.xmlで取得したＵＲＬの前に「/」+「Webアプリケーションアーカイブ形式のファイルに拡張子を除いた部分」+「/servet」という文字列が必要となる。よって本実施形態の「sample.war」では、「/sample/servlet」という文字列が必要になるのでそれを追加する。そして得られたサーブレットにアクセスすることができるＵＲＬをメモリ上に保存する。本実施形態の中から例を挙げると開始画面にアクセスするためには、「/sample/servlet/Index」という文字列を一覧のひとつとして保存しておく。またその文字列は、開発支援部301を終了するまでコンピュータのメモリ上に保存しておく。

【００６２】ステップ1002で該当WebアプリケーションにアクセスするためのＵＲＬを全て取得したら、ステップ1003からステップ1005で、ユーザがブラウザを使って該当Webアプリケーションに使用したときに、サーブレットやJavaBeansが使用するデータを格納したファイルの名称をすべて取得する。本実施形態では、ひとつのデータファイルを複数のサーブレットが使用することができるとし、全てのデータファイルを取得しているが、対象とする画面を絞り込んだり、データファイルがない場合は何も表示しなくても良い。データファイルは、例えば通常のコンピュータに保存されているファイルや、データベースに格納されているデータなどで良く、どちらも容易に、対象とするファイルの一覧を取得することができる。

【００６３】まず、ステップ1003で、送信データ保存装置308から、データを格納しているデータファイルの名称を取得する。本実施形態の場合、ステップ406で開発支援部301が初めて起動されたときに、送信データ保存装置308にはデータファイルが保存されていないので何も取得しない。

【００６４】次に、ステップ1004で、クライアント側データ保存装置309から、データを格納しているデータファイルの名称を取得する。本実施形態の場合、ステップ406で開発支援部301が初めて起動されたときに、クライアント側データ保存装置309にはデータファイルが保存されていないので何も取得しない。

【００６５】続いて、ステップ1005で、サーバ側データ保存装置310から、データを格納しているデータファイルの名称を取得する。本実施形態の場合、ステップ406で開発支援部301が初めて起動されたときに、サーバ側データ保存装置310には何も保存されていないので、何も取得しない。

【００６６】ステップ1005が終了し、ステップ1006に進むと、ステップ1002で取得したWebアプリケーションにアクセスするためのＵＲＬの一覧と、ステップ1003からステップ1005で取得した各データファイルの一覧を、画面選択装置305が端末304に表示する。なお、ＵＲＬの一覧はプルダウンリストボックスの形式で表示する。

【００６７】図１１は、図１０で詳細に示したステップ406の処理が終了したとき、端末304に表示される画面例を示す。

【００６８】アプリケーション開発者303が、Webアプリケーションとして提供されている開発支援部301を起動するために使用したブラウザ1101に、当該開発支援部301を起動するためのＵＲＬと該当Webアプリケーションを指定するための引数を与えたＵＲＬ1102を指定する。しばらく待って、図１０で説明した開発支援部301の処理が終了すると、該当Webアプリケーションが公開している画面の一覧を参照して選択する領域1103を表示する。本発明に係る開発支援部301が始めて起動されたので、データファイルの一覧と、選択領域はまだ表示されない。

【００６９】ステップ406が終了し、画面にＵＲＬおよびデータファイル名の各一覧が表示されたら、ステップ407に進む。図１２を使って、アプリケーション開発者303が行うステップ407からステップ409の処理を説明する。

【００７０】まず、ステップ407で、アプリケーション開発者303がブラウザ上にあるＵＲＬの一覧から、表示させたい画面、すなわちアクセスしたいＵＲＬを選択する。本実施形態の場合、Index 1201を選択している。

【００７１】次に、ステップ408では、アプリケーショ

ン開発者303が、データファイルの一覧からステップ407で選択したサーブレットやそのサーブレットが呼び出すJavaBeansが使用するデータを保存しているデータファイルの名称を選択する。本実施形態の図１２では、データファイルが全くないので何も選択していない状態になる。

【００７２】ステップ409では、アプリケーション開発者303が、ＵＲＬとデータファイルを選択したあと、遷移を実行する「実行ボタン」1202を押す。「実行ボタン」1202が押されると、開発支援部301は、Webサーバ上のステップ407で選択された該当WebアプリケーションのＵＲＬにアクセスする。そのときにそのＵＲＬでのWebアプリケーションの業務処理で使用されるデータを読み込むためのデータファイルの名称も送信する。本実施形態では、データファイルの名称をキーワードのひとつとする。このキーワードは、送信データ保存装置308、クライアント側データ保存装置309、及びサーバ側データ保存装置310のそれぞれに格納するデータファイルのデータファイル名称であり、かつ送信データ保存装置308、クライアント側データ保存装置309、及びサーバ側データ保存装置310に格納されているデータを取得するため、それらのデータが格納されているデータファイルを取得するキーワードとするものである。本実施形態の場合、該当WebアプリケーションのアクセスしたいサーブレットはindexServletであり、ステップ407で選択されたＵＲＬ「/sample/servlet/Index」にアクセスする。そのとき同時に、データファイルに対応するキーワードをDATAFILEとすると、選択されたデータファイルの名称がないので、DATAFILE=と言う形をブラウザが作成して送信する。

【００７３】次に、要求を受け取った開発支援部301がWebアプリケーションが使用する領域にデータを設定し、サーブレットが業務処理を行い、開発支援部301がデータファイルを生成する処理を説明する。

【００７４】まず、図１３を用いて、図４のステップ409によって送信されたキーワードとデータを受信した該当Webアプリケーションが開発支援部301と連携して行う処理の詳細を説明する。

【００７５】ステップ1301で、Webサーバにある開発支援部301は、ステップ408で選択された、該当Webアプリケーションが業務処理で使用するデータファイルの名称を受信する。

【００７６】このとき、開発支援部301がWebアプリケーションであり、ブラウザから要求を出しているので、これらの要求を該当サーブレットが取得することは、キーワードとそのキーワードに対応する値を対にして送信する一般のWebアプリケーションと同じ手法で可能である。本実施形態では、開発支援部301が取得するデータファイルに対応するキーワードをDATAFILEとしているので、ステップ407で選択されたサーブレットIndexは、データファイルの名称として、長さ0の空文字を受け取る。実際には、ファイル名に空文字は使用できないのでデータファイルを取得しないことになる。

【００７７】ステップ1301が終了したら、ステップ1302で、送信データ保存装置308、クライアント側データ保存装置309、及びサーバ側データ保存装置310に保存されているデータを取得し、該当Webアプリケーションが使用する領域に設定するために、データ設定装置314を起動する。データ設定装置314は、ＵＲＬ･データ対応表を生成して、データの種類に該当する領域にデータを設定する機能を持っている。

【００７８】図１４は、ＵＲＬ･データ対応表を作成する処理の手順を示すフローチャートである。図１４を使って、データを格納する際のファイルの名称であり、また取得するためのキーワードであるデータファイルの名称とデータの種類を対応付けるＵＲＬ･データ対応表を作成する処理の流れを説明する。

【００７９】ステップ1302で起動されたデータ設定装置314は、ステップ1401で、ＵＲＬ･データ対応表がメモリ上に存在するかを調べる。データ設定装置314が初めて起動されたときはＵＲＬ･データ対応表がないので、ＵＲＬ･データ対応表を作成する処理を行うため、ステップ1402に進む。またデータ設定装置314が起動されていたとしてもＵＲＬ･データ対応表がないときは、ステップ1402に進み、ＵＲＬ･データ対応表を作成する処理を行う。

【００８０】ステップ1402の処理では、図１０の処理により生成されメモリ上に保存されているＵＲＬの一覧を取得する。本実施形態の場合、/sample/servlet/Index、/sample/servlet/Registarion、/sample/servlet/CustomerUpdate、/sample/servlet/CustomerConfirm、/sample/servlet/ItemSelect、/sample/servlet/ItemConfirm、及び/sample/servlet/LogoutのＵＲＬを取得する。

【００８１】次に、ステップ1403では、ステップ1402で取得したＵＲＬ一覧の中で、ファイル名に使うことができない文字をファイル名に使うことができる文字に変換する。例えば、「/」をファイル名に使うことができる文字「_」に変換する処理を行う。本実施形態の場合、例えば「/sample/servlet/Index」ならば「_sample_servlet_Index」に変換する。

【００８２】次に、ステップ1404では、設定保存装置317から、ファイル名が重複しないように動的に変化する固定文字列を追加する。本実施形態の場合、「_ID????」をステップ1403で生成した文字列に追加する。従って、上述した「_sample_servlet_Index」の例では「_sample_servlet_Index_ID????」となり、データ保存装置316では保存するときに????の部分を動的生成する処理を行う。

【００８３】ステップ1405で、データファイルが、送信データを保存しているファイルなのか、クライアント側

のデータを保存しているファイルなのか、サーバ側のデータを保存しているファイルなのかを判別するために、設定保存装置317からデータファイルの種類とそれを表す文字列との対応表を読み込み、送信データ、クライアント側でのデータ、及びサーバ側のデータそれぞれに対して、それぞれキーワードとなる文字列を取得し、データファイルの名称に追加する。

【００８４】図１５は、データファイルの種類とデータファイルに含まれる文字列の対応表の一例である。データファイルの種類1501には、送信データなのか、サーバ側データなのか、クライアント側データなのかを定義する。データファイルに追加する文字列1502には、それらのデータファイルの名称にどのような文字列を付けるかを定義する。本実施形態の場合、クライアント側データファイルには「.client.ser」1503、送信データのデータファイルには「.send.ser」1504、サーバ側データのデータファイルには「.server.ser」1505の文字列を、それぞれ追加する。例えば、「_sample_servlet_Index_ID????」の後に上記説明した文字列を付け、クライアント側データファイルの名称は「_sample_servlet_Index_ID????.client.ser」、サーバ側ファイルの名称は「_sample_servlet_Index_ID????.server.ser」、サーバでの処理が終了してブラウザに送られるデータのデータファイルの名称は「_sample_servlet_Index_ID????. send.ser」となる。

【００８５】ステップ1402で取得したＵＲＬの文字列全てに対してステップ1405までの処理が終了したら、メモリ上に、ＵＲＬ･データ対応表が作成される。

【００８６】図１６は、ＵＲＬ･データ対応表の一例である。ＵＲＬ1601は、図１０の処理で取得したWebアプリケーションにアクセスするためのＵＲＬである。データファイル名1602は、ＵＲＬ1601に対応したデータファイルの名称である。図１６の場合、ＵＲＬ1601が/sample/servlet/Index1603に対応するデータファイルの名称は、_sample_servlet_Index_ID????.client.ser1604、_sample_servlet_Index_ID????.server.ser1605、及び_sample_servlet_Index_ID????. send.ser1606であること示している。また、図１５のデータファイルに付ける文字列から、_sample_servlet_Index_ID????.client.ser1604はクライアント側データのデータファイル、_sample_servlet_Index_ID????.server.ser1605はサーバ側データのデータファイル、_sample_servlet_Index_ID????.send.ser1606は送信データのデータファイルと定義される。そして、各データファイル名の????1607は、データファイル名称が重複して上書きされないように動的に変化する部分を表し、本実施形態の場合の????1607は4桁であることを示している。????の生成については後で説明する。

【００８７】次に、図１７を使ってデータファイルの構造について説明する。キーワード1701は、送信されてきたデータを識別するために使用する文字列であり、値1702は、キーワード1701に割り当てられた値である。図１７では、キーワード1701の例として、USERNAME 1703に割り当てられた値1702はscott1704である。

【００８８】また、データファイルは、図１７に示すようなキーワードとそのキーワードに対応する値を対にして持っているテキストファイルでも良いし、Webアプリケーションがデータを使用する際のコンピュータのメモリ上にあるクラスを、バイナリーデータとしてそのままファイルに保存するシリアライズといわれる機能を使って保存してもよい。前者であれば通常のテキストエディタで容易に作成可能である。また、Javaではシリアライズする機能やシリアライズされているデータを元に戻すデシリアライズという機能を、提供されているクラスを使うことで容易に実現できる。前者の場合は、サーブレットでキーワードで値を取得して、サーブレットで使用する変数に設定することができるし、後者の場合は、デシリアライズしたデータをサーブレット内で定義されている変数と置き換えたり代入することで、各データ保存装置に保存されていたデータをサーブレットで使用する変数に設定することができる。これらの形式は混在することが可能である。本実施形態の場合、図７の本発明に係るスーパークラスであるBaseServletで、サーブレットが受信するデータが格納されている変数req 715から取得し、Javaが提供するHashtableクラスに変換して(行703、行704、行705)、サブクラスで使用するように規定している。またコンピュータのメモリ上にあるクラスをバイナリーデータとしてそのままファイルに保存するシリアライズという方法を用い、Hashtable型の変数clientData、sendData、sessionDataをシリアライズして、各データ保存装置に保存する。

【００８９】再び図１４を参照して、ステップ1402からステップ1405までの処理でＵＲＬ･データ対応表ができたら、ステップ1301で取得したデータファイルの名称から、送信データ保存装置308、クライアント側データ保存装置309、及びサーバ側データ保存装置310のいずれかから、データファイルを取得して、当該データをWebアプリケーションが使用する領域に設定する処理を行う。

【００９０】まず、ステップ1406で、サーバ側データ保存装置310からデータを取得して、該当Webアプリケーションのサーブレットが業務処理を始めるより先に、サーバ側のデータ領域に設定する処理を行う。ステップ1406の処理の詳細を、図１８を使って説明する。

【００９１】ステップ1801では、該当Webアプリケーションが受信したデータファイルを取得するためのキーワードを取得する。本実施形態の場合、送信されたデータファイルの名称が空文字なので何も取得しない。

【００９２】ステップ1802では、ステップ1801で取得したデータファイルの中からサーバ側のデータを保存したデータファイルのみを取得する。サーバ側のデータを保

存したデータファイルの名称には、すでに説明したように「.server.ser」という文字列があるので、「.server.ser」という文字列のあるデータファイルがサーバ側のデータを保持しているデータファイルになる。本実施形態の場合、送信されたデータファイルの名称が空文字なのでなにも取得しない。

【００９３】ステップ1803では、ステップ1802で取得したデータファイルの名称を使って、サーバ側データ保存装置310からサーバ側データを取得する。本実施形態の場合、データファイル名称を空文字で取得しているのでファイルを取得することはない。

【００９４】ステップ1804では、ステップ1803で読み込んだデータファイルをデシリアライズする処理を行う。本実施形態では、データファイルの名称が空文字なのでデシリアライズはしない。

【００９５】ステップ1805では、ステップ1804で復元されたデータを現在メモリ上にあるセッションのデータとして扱えるようにする処理を行う。この処理では、サーバ側データのファイルがシリアライズされたとき、メモリ上にあるセッション用の変数をそのままシリアライズしてあれば、現在メモリ上にあるセッション用の変数にデシリアライズされたセッション用の変数を代入することで入れかえることができる。また、データファイルを取得していないためデシリアライズしない場合は、アプリケーションサーバが持っているセッション315の値を取得する。本実施形態の場合、データファイルの名称に空文字が送られたためデータファイルを取得せず、デシリアライズもしなかったため、アプリケーションサーバにあるセッション315に保存されているキーワードとデータのペアを取得して、セッション用の変数に代入する。開始画面の場合、該当Webアプリケーションのスタートなので、セッションが生成されるためデータは格納されていない。

【００９６】ステップ1805が終了すると、図１４に戻って、ステップ1407へ進む。ステップ1407では、該当Webアプリケーションにアクセスしたときに、サーブレットが業務処理で使用するクライアント側データを、サーブレットのクライアント用の変数に設定する処理を行う。ここで、クライアント側データとは、ユーザが入力する値や、ユーザ情報を保存したりするブラウザが持っているクッキーという機能で得られるキーワードとそのキーワードに対応する値を対にしたデータである。例えば、検索サービスを行っているWebアプリケーションの場合、検索条件がこれに相当する。図１９を使って、ステップ1407の処理を説明する。

【００９７】まず、ステップ1901では、送信されたデータファイルを取得する。このとき、送信されるのはデータファイルの名称だけである。本実施形態の場合、送信されたデータファイルの名称が空文字を取得する。

【００９８】次に、ステップ1902では、ステップ1901で取得したデータファイルの中からクライアント側データを保存したデータファイルのみを取得する。クライアント側のデータを保存したデータファイルの名称には、すでに示したように「.client.ser」という文字列があるので、「.client.ser」という文字列のあるデータファイルがクライアント側のデータを保持しているデータファイルになる。本実施形態の場合、取得したデータファイルの名称が空文字なので、有効なデータファイルの名称は取得しない。

【００９９】ステップ1903では、ステップ1902により取得したデータファイルの名称を使って、クライアント側データ保存装置309からクライアント側データを取得する。本実施形態の場合、有効なデータファイルの名称を取得していないので、データファイルを取得しない。

【０１００】ステップ1904では、ステップ1903で読み込んだデータファイルをデシリアライズする処理を行う。本実施形態では、ユーザが入力したデータをキーワードとそのキーワードに対応した形で保存するので、クライアント側データファイルをデシリアライズすれば、クライアント側データ保存装置309に保存されているデータを復元することができる。本実施形態の場合、図７のBaseServletで定義されている処理でHashtable clientData = getClientData(req);がこの処理を行う。本実施形態の場合、データファイルを取得していないので、ブラウザが送信した、ユーザが入力した情報やクッキーの情報を取得する。

【０１０１】ステップ1905では、ステップ1904で復元されたデータを、サーブレットがあたかもユーザが入力したデータとして扱えるようにする処理を行う。本実施形態の場合、サブクラスが、クライアントデータを取得するときは、clientDataから取得するので、取得したデータをclientDataに設定することで、サーブレットはブラウザから送信されたデータと同じように扱える。

【０１０２】ステップ1905の処理が終了したら、図１４に戻り、図１４の処理を終了する。ここまでで図１３のステップ1302が終了したこととなり、次にステップ1303に進む。

【０１０３】ステップ1303までの処理で、該当Webアプリケーションがセッションに格納されているデータを使う場合でも、ブラウザを使っているユーザが入力した値を使う場合でも、すでにサーブレットが通常の処理でアクセスできるようになっているので、サーブレットはそれらの値を取得することができる。従って、ステップ1303では、ステップ407で選択されたサーブレットが必要に応じてセッション用の変数クライアント用の変数を使って、セッション315に格納されているデータやユーザが入力したデータを取得し、業務処理を進めることができる。本実施形態の場合、図８に示している操作doBusiness()803がこの処理を行う。アプリケーション開発者303が、Webアプリケーションの設計工程で作成される仕

様に基づきdoBusiness()803を実装する。

【０１０４】ステップ1304では、ステップ1303でサーブレットが処理を行いブラウザに送信するデータを生成したら、その生成したデータをキーワードとそのキーワードに対応する値とを対にして設定して行く。キーワードはWebアプリケーションの設計工程で導出されたものである。本実施形態の場合、図８に示したsendData.put("USERNAME", userName);806と、sessionData.put("USERNAME", useName);807が、その処理を行う。

【０１０５】まず、送信データ用の変数sendDataにキーワードUSERNAMEに対応する値として、userNameを設定する。次に、セッション用の変数sessionDataにも、同様に、キーワード USERNAMEに対応する値として、userNameを設定する。本実施形態では、送信データ用の変数を設けることによりブラウザに送信するデータで動的な部分のみを取得することができるので、それを、送信データとして保存することもできるし、ブラウザに表示するHTML形式のデータを文字列（行806）として保存して、例えばキーワードとしてHTMLと対応付け、そのキーワードに対応する値としてHTML形式のデータとして保存することもできる。送信データは、アプリケーション開発者がテスト工程で正しいデータが送られているかを確認するために使用することもできる。

【０１０６】ステップ1304で、アプリケーション開発者303が実装段階で、送信データ保存装置308、及びサーバ側データ保存装置310に保存するデータを、スーパークラスBaseServletの変数sendData、sessionDataに設定したら、ステップ1305で、開発支援部301は、それらに設定されているデータを各保存装置に設定する処理を行う。本実施形態の場合、図７に示しているsetClientData(clientData);711と、setSendData(sendData);712と、setSessionData(sessionData);713が、それぞれ、クライアント側データ、送信データ、サーバ側データを保存する処理を行う。図２０を使って、これらの処理の詳細を説明する。

【０１０７】まず、ステップ2001で、保存するデータの種類を決定する。本実施形態の場合、保存するデータごとに操作を分けているので、この操作を分けることが該当する。説明のためにクライアント側データを保存する処理を例にあげて説明する。

【０１０８】ステップ2001で保存するデータの種類を決定したら、ステップ2002及び2003で、保存装置に保存するデータファイルの名称を取得する。本実施形態の場合、setClientData()では、クライアント側データを保存するのでクライアント側データ保存装置309に保存するためのファイル名を生成する。メモリ上にあるＵＲＬ･データファイル対応表とサーブレットのクラス名に対応する送信データを保存するためのファイルを取得する。

【０１０９】まずステップ2002で、画面名とサーブレットとＵＲＬの対応表から、サーブレットに対応したＵＲＬを取得する。本実施形態の場合、IndexServletの例を示しているので、図６のサーブレット602からIndexServlet606に対応しているＵＲＬ603である/Index 605を取得する。

【０１１０】ステップ2002で該当Webアプリケーションにおいてアプリケーションサーバに依存しない部分のＵＲＬの文字列を取得したら、ステップ2003で、ＵＲＬ･データ対応表のＵＲＬの中で、ステップ2002で取得した文字列を一番最後に持っているＵＲＬを取得し、データファイルの種類とそれを表す文字列のデータファイルに付ける文字列とを使って、それに対応しているデータファイル名を取得する。本実施形態の場合、ステップ2002で/Indexという文字列を取得しているので、ＵＲＬ･データ対応表のＵＲＬから、/Index605を含んでいるＵＲＬ /sample/servlet/Index 1603を取得し、そのＵＲＬに対応しているデータファイル名_sample_servlet_Index_ID????.client.ser、_sample_servlet_Index_ID????.send.ser、_sample_servlet_Index_ID????.server.serから、クライアント側データのファイル名を表す.client.ser1503を含んでいる文字列_sample_servlet_Index_ID????.client.serを取得する。

【０１１１】ステップ2003で保存するデータファイル名が取得できたら、ステップ2004で、ファイル名が重複しないように、ステップ2003で取得した文字列の中の????の部分を置換する。????には、データ保存装置に保存されているデータファイル全ての中から????の部分が一番大きいものを取得する。これは正規表現を用いることで簡単に実現可能である。例えば、Windows（マイクロソフト社製のOS）のファイルシステムに格納されているときは、「dir /ON _sample_servlet_Index_ID*.client.ser」を実行すると、????の部分のみが異なるファイルの一覧が昇順に並べ替えられて取得できるので、????の部分に「1」を足したものを????とする。本実施形態の場合、図１０の処理により、保存されているデータファイル名を全て取得しメモリ上に保存しているが、データファイルの名称は何も取得していないので、????部分には0001を、ファイル名が重複しないようにするために付ける。最終的に生成されるクライアント側データ保存ファイルの名称は、_sample_servlet_Index_ID0001.client.serとなる。

【０１１２】ステップ2004でクライアント側データ保存装置309に保存するデータファイルの名称を決定したので、ステップ2005では、処理結果の変数をシリアライズする処理を行う。本実施形態の場合、setClientData(clientData);の引数に指定されるclientDataがクライアント側データ用の変数として使われているのでこれをシリアライズする。ブラウザから送信されるデータはないので、開発支援部301が始めて起動されたためデータは保存されない。

【０１１３】最後にステップ2006で、ステップ2005で作成されたクライアント側データのデータファイルをクライアント側データ保存装置309に保存する。本実施形態の場合、ステップ2005でシリアライズしたデータをデータファイル名_sample_servlet_Index_ID0001.client.serとして、クライアント側データ保存装置309に保存する。

【０１１４】ステップ2006でクライアント側データを保存したら、ステップ2007に進み、クライアント側データ、サーバ側データ、及び送信データの全てのデータの種類に対してデータを保存したかどうか判定する（ステップ2007）。全データ種類について保存したら、図２０の処理を終了する。

【０１１５】本実施形態の場合、図２０の処理により、クライアント側データのデータファイル_sample_servlet_Index_ID0001.client.serには、入力した値、例えば、ブラウザに対してユーザID 「abcde」を入力したら、キーワード「USERID」に対する値として「abcde」が保存される。しかし、現段階では、何も入力されていないので、何もデータが保存されない。また、送信データのデータファイル_sample_servlet_Index_ID0001.send.serには、行806で設定したキーワード「USERNAME」に対応する値として、業務処理行805で取得したユーザ名、例えばscottが保存される。しかし、現段階では、何も入力されていないのでデータ保存されない。

【０１１６】また、サーバ側データのデータファイル_sample_servlet_Index_ID0001.server.serには、図８の行807で設定したキーワード「USERNAME」に対応する値として、業務処理行805で取得したユーザ名、例えばscottが保存される。しかし、現段階では、何も入力されていないのでデータ保存されない。

【０１１７】図２０の処理が終了したということは、図１３のステップ1305まで処理したということである。ステップ1305の後、ステップ1306に進み、サーブレットの業務処理で作成されたHTML形式のデータをブラウザに送信し、ブラウザが受信したHTML形式のデータを表示する。本実施形態の場合、図８の行808の処理で作成されたHTML形式のデータを行810で返却し、本発明に係るスーパークラスであるBaseServletの行714で上記処理を行い、図２１の画面をブラウザに表示し一連の流れは終了する。

【０１１８】ここまでの処理で、各データファイルがそれぞれひとつずつ作成されて、各保存装置に保存され、図２１に示す開始画面を表示することができる。現段階では、データの入力が何もないのと等しいので、該当Webアプリケーションを起動したことと同じ状態でブラウザに表示される。

【０１１９】次に、先に説明したデータファイルのうち、送信データファイルとサーバ側データファイルに値が保存されているときに、アプリケーション開発者303により再びステップ406の処理が実行され、本発明に係る開発支援部301が起動された場合を説明する。送信データ及びサーバ側データとして保存されているデータは、先に説明したキーワード「USERNAME」に対する値である「scott」とする。

【０１２０】ここで、BaseServletを継承して作成するサブクラスの実装について説明する。図２２は、サーブレットのサブクラスの実装の一例である。

【０１２１】行public class ItemSelectServlet extends BaseServlet2201の記述では、本実施形態で提供しているサーブレットのスーパークラスBaseServletを継承して、ログイン処理を行う商品選択画面のサーブレットのクラスItemSelectServlet607を定義している。そして、このサーブレットを起動するためのURLでアプリケーションサーバに依存しない部分の文字列は/ItemSelect608である。

【０１２２】doBusiness();には、すでに説明したように、ItemSelectServletの業務処理を記述している。本実施形態の商品選択画面では、開始画面から遷移する画面のひとつで、セッションに格納されているユーザ名を取得して、商品のリストと共に表示する処理を行う。このとき、セッションにユーザ名がないときは、不正なアクセスとしてその旨を表す表示をするという処理も行う。

【０１２３】String userName = (String)sessionData.get("USERNAME");2202は、セッションに格納されているキーワード「USERNAME」に対応するユーザ名を取得する処理である。また、行2203は、セッションからキーワード「USERNAME」が取得できないときに行う処理である。

【０１２４】String itemList= getItemList(); 2204は、データベースから商品一覧を取得して、itemListに代入する処理である。操作getItemList()2204は、クラスItemSelectServlet内で定義され、操作getItemList()2204の処理では、JavaBeansを使って、データベースにアクセスして商品一覧を取得して、HTML形式に整形する。

【０１２５】sessionData.put("USRENAME", userName);2205は、行2202で取得したユーザ名を、サーバ側データとして、セッションに保存するために、キーワード「USERNAME」を対応付けて格納する処理である。これら、業務処理に対応付けるキーワードは、Webアプリケーションの設計工程で、アプリケーション開発者303が定義するものである。すでに説明したキーワードとデータとデータの種類の対応表を使う。

【０１２６】行2206は、ブラウザにユーザ名と商品を表示するためのHTML形式のデータを生成する処理である。本実施形態の場合、動的に変更して表示するデータのユーザ名を、行705の業務処理で取得しているので、ユーザ名の変数userNameをHTML形式のデータに埋め込み、行

2204で取得した商品一覧のHTML形式の文字列を作成している。そして、return htmlStream;で、作成したHTML形式のデータを返却している。

【０１２７】ステップ406で、再び該当Webアプリケーションを指定して本発明に係る開発支援部301を起動したとする。すでに説明したように、開発支援部301を起動するサーブレットのＵＲＬに、引数として選択されたキーワードを付け加えて、例えば「http://localhost:8080/Index?WAR=c%3A%5Cproject%5Csample.war」としWebアプリケーションを特定して起動する。ステップ406により起動された開発支援部301は、すでに説明したように、図１０の処理を行う。

【０１２８】まず、すでに説明したように、ステップ1001で、指定された該当Webアプリケーションアーカイブファイルに格納されているファイルの中からweb.xmlを取得する。次に、ステップ1002で、そのファイル内を解析して、ブラウザからWebアプリケーションにアクセスするためのＵＲＬを取得する。ステップ1002で該当WebアプリケーションにアクセスするためのＵＲＬを全て取得したら、すでに説明したように、ステップ1003からステップ1005で、ユーザがブラウザを使って該当Webアプリケーションを起動したときに、サーブレットやJavaBeansが使用するデータを格納したファイルの名称をすべて取得する。

【０１２９】本実施形態では、ひとつのデータファイルを複数のサーブレットが使用することができるとし、全てのデータファイルを取得しているが、対象とする画面を絞り込んで、データファイルがない場合は何も表示しなくても良い。データファイルは、例えば通常のコンピュータに保存されているファイルや、データベースに格納されているデータなどで良く、どちらも容易に対象とするファイルの一覧を取得することができる。

【０１３０】まず、ステップ1003で、送信データ保存装置308から、データを格納しているデータファイルの名称を取得する。本実施形態の場合、先に起動されたときに、_sample_servlet_Index_ID0001.send.serが作成され、送信データ保存装置308に保存されているので、このデータファイル名を取得しメモリ上に保存しておく。

【０１３１】次に、すでに説明したように、ステップ1004で、クライアント側データ保存装置309から、データを格納しているデータファイルの名称を取得する。本実施形態の場合、ステップ405で開発支援部301が起動されたときに、クライアント側データ保存装置309には、データファイル_sample_servlet_Index_ID0001.client.serが格納されているので、このデータファイル名を取得しメモリ上に保存しておく。

【０１３２】次に、ステップ1005で、サーバ側データ保存装置310から、データを格納しているデータファイルの名称を取得する。本実施形態の場合、先に起動されたときに、サーバ側データ保存装置310に、データファイルとして_sample_servlet_Login_ID0001.server.serが格納されているので、このデータファイル名を取得しメモリ上に保存しておく。

【０１３３】ステップ1005が終了しステップ1006に進むと、ステップ1002で取得したブラウザを通じてWebアプリケーションにアクセスするためのＵＲＬの一覧と、ステップ1003からステップ1005で取得した各データファイルの一覧を、画面選択装置305が端末304に表示する。

【０１３４】図１０の処理が終了し、ステップ406の処理が終了すると、図２３の画面が端末304に表示される。

【０１３５】以上のようにして、アプリケーション開発者303が開発支援部301を起動すると、すでに説明した図１０の処理を行い、その処理が終了すると、図１１で説明した該当Webアプリケーションが公開している画面の一覧を参照して選択する領域と、図２３に示したデータファイルの一覧を参照する領域2301と選択する領域2302が表示される。ここで、データファイルの名称を取得する際に内容も取得しておけば、データファイルに格納されている情報も同時に表示することもできるが、煩雑になるので、本実施形態の場合、各保存装置からデータを取得するためのキーワードとなるデータファイルの名称だけを表示するものとする。

【０１３６】ステップ406が終了し、図２３のように画面にＵＲＬおよびデータファイル名の各一覧が表示されたら、ステップ407に進む。アプリケーション開発者303が行うステップ407からステップ409の処理を、図２４を使って説明する。

【０１３７】すでに説明したステップ407では、アプリケーション開発者303が、ブラウザに表示されているＵＲＬの一覧から、アクセスしたいＵＲＬ、すなわち起動したいサーブレットを選択する。本実施形態の場合、サーブレットItemSelectServletを起動するＵＲＬとして/sample/servlet/ItemSelect 2401を選択している。

【０１３８】次に、ステップ408で、アプリケーション開発者303が、データファイルの一覧から、ステップ407で選択したサーブレットやそのサーブレットが呼び出すJavaBeansが使用するデータを保存しているデータファイルの名称を選択する。データファイル選択領域2302はＵＲＬ一覧1003で選択されたＵＲＬで起動するサーブレットの処理で使用するデータをアプリケーション開発者が選択する部分であり、選択すると「レ」2402が表示される。本実施形態では、3つのデータファイル_sample_servlet_Login_ID0001.client.serと、_sample_servlet_Login_ID0001.send.serと、_sample_servlet_Login_ID0001.server.serを選択している。

【０１３９】ステップ408が終了したら、ステップ409に進み、アプリケーション開発者303が、ＵＲＬとデータファイルを選択したあと、遷移を実行する「実行ボタン」2403を押す。「実行ボタン」2403が押されると、開

発支援部301は、Webサーバ上のステップ407で選択された該当WebアプリケーションのＵＲＬにアクセスする。そのときに、そのＵＲＬでのWebアプリケーションの業務処理で使用されるデータを読み込むためのデータファイルの名称も送信する。

【０１４０】本実施形態の場合、該当Webアプリケーションで起動させるサーブレットはItemSelectServletであり、ステップ407で選択されたＵＲＬ「/sample/servlet/ItemSelect」にアクセスする。そのとき同時に、データファイルに対応するキーワードをDATAFILEとすると、選択されたデータファイルの名称を付けてブラウザがDATAFILE=_sample_servlet_Index_ID0001.client.ser&;DATAFILE=_sample_servlet_Index_ID0001.send.ser&;DATAFILE=_sample_servlet_Index_ID0002.server.serという式を作成し、要求として送信する。本実施形態では、データファイルの名称をキーワードのひとつとし、送信データ保存装置308、クライアント側データ保存装置309、及びサーバ側データ保存装置310それぞれに格納するデータファイルのデータファイル名称であり、かつ送信データ保存装置308、クライアント側データ保存装置309、及びサーバ側データ保存装置310に格納されているデータを取得するため、データが格納されているデータファイルを取得するキーワードとする。

【０１４１】次に、要求を受け取った開発支援部301がWebアプリケーションが使用する領域にデータを設定し、サーブレットが業務処理を行い、開発支援部301がデータファイルを生成する、という処理を説明する。

【０１４２】まず、図４のステップ409によって送信されたキーワードとデータを受信した該当Webアプリケーションが、開発支援部301と連携して行う処理の詳細を、図１３を使って説明する。

【０１４３】ステップ1301で、Webサーバにある本発明に係る開発支援部301は、ステップ408により選択された該当Webアプリケーションが業務処理で使用するデータファイルの名称を受信する。

【０１４４】このとき、開発支援部301がWebアプリケーションであり、ブラウザから要求を出しているので、これらの要求を該当サーブレットが取得することは、キーワードとそのキーワードに対応する値を対にして送信する一般のWebアプリケーションと同じ手法で可能である。本実施形態では、開発支援部301が取得するデータファイルに対応するキーワードをDATAFILEとしているので、ステップ407で選択されたアクセスしたいサーブレットItemSelectServletは、データファイルの名称として、_sample_servlet_Index_ID0001.client.serと、_sample_servlet_Index_ID0001.send.serと、_sample_servlet_Index_ID0001.server.serを受信する。

【０１４５】ステップ1301が終了してデータファイルの名称を取得したら、ステップ1302に進む。ステップ1302で、送信データ保存装置308、クライアント側データ保存装置309、及びサーバ側データ保存装置310に保存されているデータを取得し、該当Webアプリケーションが使用する領域に設定するために、データ設定装置314を起動する。

【０１４６】次に、ステップ1301で取得したデータファイルの名称から、送信データ保存装置308、クライアント側データ保存装置309、及びサーバ側データ保存装置310のいずれかから、データファイルを取得して、そのデータをWebアプリケーションが使用する領域に設定する処理を行う。この処理では、まず図１４で説明したフローが実施される。URL・データ対応表はすでに存在するので、ステップ1401から1406に進む。

【０１４７】ステップ1406で、サーバ側データ保存装置310から取得したサーバ側データを、該当Webアプリケーションのサーブレットが業務処理を始めるより先に、サーバ側のデータ領域に設定する処理を行う。ステップ1406の処理の詳細を、図１８を使って説明する。

【０１４８】ステップ1801では、該当Webアプリケーションが受信したデータファイルを取得するためのキーワードを取得する。本実施形態の場合、送信されたデータファイルの名称は、_sample_servlet_Index_ID0001.client.serと、_sample_servlet_Index_ID0001.send.serと、_sample_servlet_Index_ID0001.server.serの3つである。

【０１４９】ステップ1801が終了するとステップ1802に進む。ステップ1802では、ステップ1801で取得したデータファイルの中からサーバ側データを保存したデータファイルのみを取得する。サーバ側のデータを保存したデータファイルの名称には、すでに説明したように、「.server.ser」という文字列があるので、「.server.ser」という文字列のあるデータファイルがサーバ側のデータを保持しているデータファイルになる。本実施形態の場合、送信されたデータファイルの名称は、_sample_servlet_Index_ID0001.client.serと、_sample_servlet_Index_ID0001.send.serと、_sample_servlet_Index_ID0001.server.serの3つであり、「.server.ser」という文字列が含まれる_sample_servlet_Index_ID0001.server.serを取得する。

【０１５０】ステップ1802が終了すると、ステップ1803へ進む。ステップ1803では、ステップ1802で取得したデータファイルの名称を使って、サーバ側データ保存装置310からサーバ側データを取得する。本実施形態の場合、サーバ側データは、キーワードとそのキーワードに対応している値を対にして複数保存することがきるセッション用の変数をシリアライズしているので、サーバ側データ保存装置310からセッション用の変数をシリアライズしたデータファイルそのもの_sample_servlet_Index_ID0001.server.serを取得する。

【０１５１】ステップ1804では、ステップ1803で読み込んだデータファイルをデシリアライズする処理を行う。

本実施形態では、セッション用の変数をそのままシリアライズしているので、サーバ側データファイルをデシリアライズすれば、サーバ側データ保存装置310に保存されているセッション用の変数を復元することができる。本実施形態では、先に起動されたときサーバ側のデータとして保存したのは、キーワードUSERNAMEに対して値がscottなので、これを復元する。

【０１５２】ステップ1805では、ステップ1804で復元されたデータを現在メモリ上にあるセッションのデータとして扱えるようにする処理を行う。この処理では、サーバ側データのファイルがシリアライズされたとき、メモリ上にあるセッション用の変数をそのままシリアライズしてあれば、現在メモリ上にあるセッション用の変数にデシリアライズされたセッション用の変数を代入することで入れかえることができる。本実施形態の場合、セッション用の変数のデータをそのままシリアライズしているので、デシリアライズしたサーバ側データをセッション用の変数に代入だけで該当Webアプリケーションがセッション用の変数を通して扱うことができる。本実施形態では、図７のBaseServletで定義されている処理でHashtable sessionData = getSessionData(req);がこの処理を行い、先にデシリアライズされたキーワードUSERNAMEに対応する値としてscottをセッション用の変数sessionDataに格納する。

【０１５３】ステップ1805が終了すると、図１４に戻って、ステップ1407へ進む。ステップ1407では、該当Webアプリケーションにアクセスするときに、サーブレットが業務処理で使用するクライアント側データを、サーブレットのクライアント用の変数に設定する処理を行う。ステップ1407の処理を、図１９を使って説明する。

【０１５４】まず、ステップ1901では、送信されたデータファイルを取得する。このとき、送信されるのはデータファイルの名称だけである。本実施形態の場合、送信されたデータファイルの名称は、_sample_servlet_Index_ID0001.client.serと、_sample_servlet_Index_ID0001.send.serと、_sample_servlet_Index_ID0001.server.serの3つであり、ステップ1901で取得するのはこれら3つのファイル名である。

【０１５５】次に、ステップ1902では、ステップ1901で取得したデータファイルの中からクライアント側データを保存したデータファイルのみを取得する。クライアント側のデータを保存したデータファイルの名称には、すでに示したように「.client.ser」という文字列があるので、「.client.ser」という文字列のあるデータファイルがクライアント側のデータを保持しているデータファイルになる。本実施形態の場合、送信されたデータファイルの名称は、_sample_servlet_Index_ID0001.client.serと、_sample_servlet_Index_ID0001.send.serと、_sample_servlet_Index_ID0001.server.serの3つであり、「.client.ser」という文字列が含まれる_sample_servlet_Index_ID0001.client.serを取得する。

【０１５６】ステップ1903では、ステップ1902により取得したデータファイルの名称を使って、クライアント側データ保存装置309からクライアント側データを取得する。本実施形態の場合、クライアント側データ用の変数をシリアライズしたので、クライアント側データ保存装置309から、クライアント側データをシリアライズしたデータファイルそのもの_sample_servlet_Index_ID0001.client.serを取得する。

【０１５７】ステップ1904では、ステップ1903で読み込んだデータファイルをデシリアライズする処理を行う。本実施形態では、ユーザが入力したデータをキーワードとそのキーワードに対応した形で保存するので、クライアント側データファイルをデシリアライズすれば、クライアント側データ保存装置309に保存されているデータを復元することができる。本実施形態の場合、図７のBaseServletで定義されている処理でHashtable clientData = getClientData(req);がこの処理を行う。ステップ1903で取得したデータファイル_sample_servlet_Index_ID0001.client.serのもとのデータが、先に起動されたときに、クライアント側データを保存していないので、クライアント側データファイル_sample_servlet_Index_ID0001.client.serからデータは復元されない。本実施形態では、開発支援部301を通して開始画面を起動したためにクライアント側データは復元されないが、開発支援部301を通さずに起動した場合、すでに説明した図１９の処理により、クライアント側データが保存され、このステップでデータが復元される。

【０１５８】ステップ1905では、ステップ1904で復元されたデータを、サーブレットがあたかもユーザが入力したデータとして扱えるようにする処理を行う。本実施形態の場合、サブクラスが、クライアントデータを取得するときは、clientDataから取得するので、復元されたデータをclinetDataに設定することで、サーブレットはブラウザから送信された通常のデータと同じようにすることができる。

【０１５９】ステップ1905の処理が終了したら図１４に戻り、図１４の処理を終了する。ここまでで図１３のステップ1302が終了したこととなり、次にステップ1303に進む。

【０１６０】ステップ1303までの処理で、該当Webアプリケーションがセッションに格納されているデータを使う場合でも、ブラウザを使っているユーザが入力した値を使う場合でも、すでにサーブレットが通常の処理でアクセスできるようになっているので、サーブレットはそれらの値を取得することができる。従って、ステップ1303で、ステップ407で選択されたサーブレットが必要に応じてセッション用の変数クライアント用の変数を使って、セッションに格納されているデータやユーザが入力したデータを取得し、業務処理を進めることができる。

本実施形態の場合、図８に示している操作doBusiness()がこの処理を行う。doBusiness()は、Webアプリケーションの設計工程で作成される仕様に基づき実装される。

【０１６１】ステップ1304では、ステップ1303でサーブレットが処理を行いブラウザに送信するデータを生成したら、その生成したデータをキーワードとそのキーワードに対応する値とを対にしたそれぞれの変数に設定する。キーワードはWebアプリケーションの設計工程で導出されたものである。本実施形態の場合、図２２に示したsessionData.put("USERNAME", useName);2205が、その処理を行う。セッション用の変数sessionDataに、キーワード USERNAMEに対応する値として、userNameを設定する。本実施形態では、送信データ用の変数を設けることによりブラウザに送信するデータで動的な部分のみを取得することができるので、それを、送信データとして保存することもできるし、ブラウザに表示するHTML形式のデータを文字列（行2206）として保存して、例えばキーワードとしてHTMLと対応付け、そのキーワードに対応する値としてHTML形式のデータとして保存することもできる。送信データは、アプリケーション開発者303がテスト工程で正しいデータが送られているかを確認するために使用することもできる。

【０１６２】ステップ1304で、アプリケーション開発者303が実装段階で、送信データ保存装置308、クライアント側データ保存装置309、及びサーバ側データ保存装置310に保存するデータを、本発明に係るスーパークラスBaseServletの変数sendData、serverDataに設定したら、ステップ1305で、開発支援部301は、それらに設定されているデータを各保存装置に設定する処理を行う。本実施形態の場合、図７に示しているsetClientData(clientData);と、setSendData(sendData);と、setSessionData(sessionData);が、それぞれ、クライアント側データ、送信データ、サーバ側データを保存する処理を行う。この処理の説明は、図２０を使ってすでに説明したので、ここでは省略する。

【０１６３】図２０の処理が終了したら、ステップ1305まで終了したということであるから、ステップ1306に進み、サーブレットの業務処理で作成されたHTML形式のデータをブラウザに送信し、ブラウザが受信したHTML形式のデータを表示する。本実施形態の場合、図２２の行2206の処理で作成されたHTML形式のデータを返却し、開発支援部301が提供しているスーパークラスであるBaseServletの行714で上記処理を行い、図２５の画面をブラウザに表示し、一連の流れは終了する。

【０１６４】図２５において、scott2501は、セッション用のデータから取得した値で、本発明に係る開発支援部301により保存されていたものである。

【０１６５】なお、本発明では、クライアントから送られる部分とセッションに保存する部分、及びクライアントに送信する部分を生成することができればよい。もし、テストや仕様変更などにより、商品選択へ到達する途中の画面、開始画面が修正中でも、クライアント側データ、サーバ側データ、及び送信データを使って対象サーブレットを起動するＵＲＬにアクセスすることができる。

【０１６６】

【発明の効果】以上説明したように、本発明によれば、不正なアクセスとみなされて、アクセスを拒否されるような画面遷移の途中にある画面であっても、プログラムを修正することなく、直接アクセスすることができるようになる。また、そのとき、前の画面から受け継がれる情報やユーザが入力した情報を、あらかじめデータファイルとして保存しておき、アクセスされたWebアプリケーションが処理を行う前に設定したり差し替えたりすることができ、これにより、アクセスされたWebアプリケーションは正常に処理を行うことができる。

【０１６７】また、Webアプリケーションの処理結果を保存することで、処理結果の確認が容易にできるようになる。そのうえ、その処理結果を、画面遷移に沿って次の画面でもデータファイルとして使用できるので、改めてデータファイルを作成する必要がなくなる。

【０１６８】さらに、開発途中のため、画面遷移の開始画面でなくWebアプリケーションの実行中にエラーが発生したときの修正は、対象としている画面に直接アクセスすることで、少ない工数で繰り返し行うことが可能である。

【図面の簡単な説明】

【図１】開発対象とするWebアプリケーションの構成の一例を示す図である。

【図２】キーワードとデータとデータの種類の対応表（開始画面の場合）の一例を示す図である。

【図３】本発明のWebアプリケーション開発支援装置を適用したシステムの構成を示す図である。

【図４】本発明を用いてWebアプリケーション開発を行った場合のフローチャートである。

【図５】Webアプリケーションでの画面遷移の一例を示す図である。

【図６】画面名とサーブレットとＵＲＬの対応表の一例を示す図である。

【図７】サーブレット（スーパークラス）の実装の一例を示す図である。

【図８】サーブレット（サブクラス）の実装の一例を示す図である。

【図９】web.xmlの一例を示す図である。

【図１０】Webアプリケーションが公開しているＵＲＬ一覧とデータファイルの一覧を取得し表示する処理の流れを示すフローチャートである。

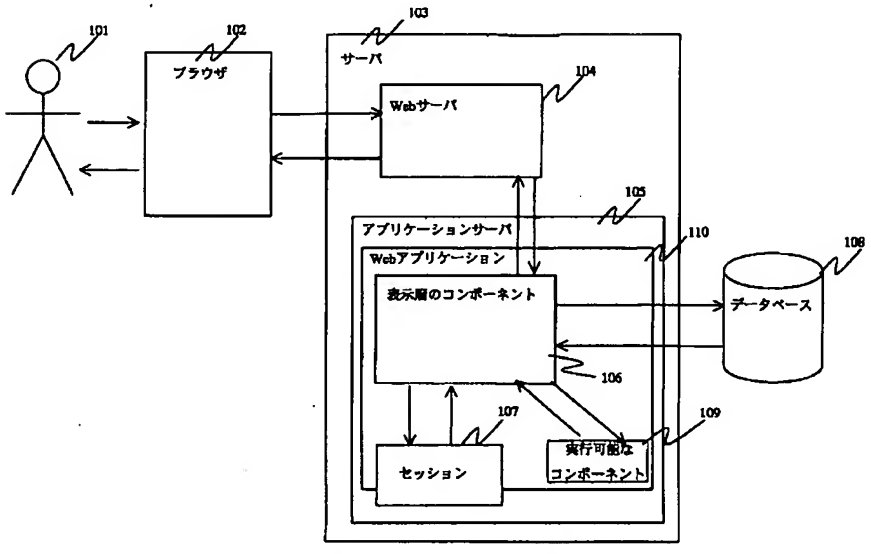
【図１１】本発明のWebアプリケーション開発支援装置を始めて起動したときの画面の一例を示す図である。

【図１２】ＵＲＬを選択する画面の一例を示す図であ

る。
【図１３】本発明の画面を表示するまでの処理の流れを示すフローチャートである。
【図１４】ＵＲＬ・データ対応表生成とデータ設定の処理の流れを示すフローチャートである。
【図１５】データファイルの種類とそれを表す文字列との対応表の一例を示す図である。
【図１６】ＵＲＬ・データ対応表の一例を示す図である。
【図１７】データファイルの構造の一例を示す図である。
【図１８】セッションに値を設定する処理の流れを示すフローチャートである。
【図１９】クライアント側データを設定する処理の流れを示すフローチャートである。
【図２０】データを保存する処理の流れを示すフローチャートである。
【図２１】サーバから返却されたHTML形式データを表示した一例を示す図である。
【図２２】サーブレット（サブクラス）の実装の一例を示す図である。
【図２３】二回目以降に実行されたときの画面の一例を示す図である。
【図２４】ＵＲＬとデータファイルを選択する画面の一例を示す図である。
【図２５】Webアプリケーションの処理によってブラウザに表示された画面の一例を示す図である。

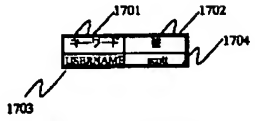
【符号の説明】
１０１…ユーザ
１０２…ブラウザ
１０３…サーバ
１０４…Webサーバ
１０５…アプリケーションサーバ
１０６，３１２…表示層のコンポーネント
１０７，３１５…セッション
１０８…データベース
１０９…実行可能なコンポーネント
１１０，３０２…Webアプリケーション
３０１…Webアプリケーション開発支援装置
３０３…アプリケーション開発者
３０４…端末
３０５…画面選択装置
３０６…データ取得装置
３０７…画面一覧取得装置
３０８…送信データ保存装置
３０９…クライアント側データ保存装置
３１０…サーバ側データ保存装置
３１１…画面一覧保存装置
３１３…キーワード送信装置
３１４…データ設定装置
３１６…データ保存装置
３１７…設定保存装置

【図１】



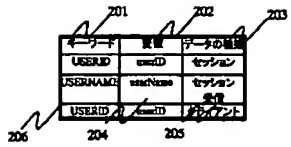
対象とするWebアプリケーションの構成一例

【図１７】



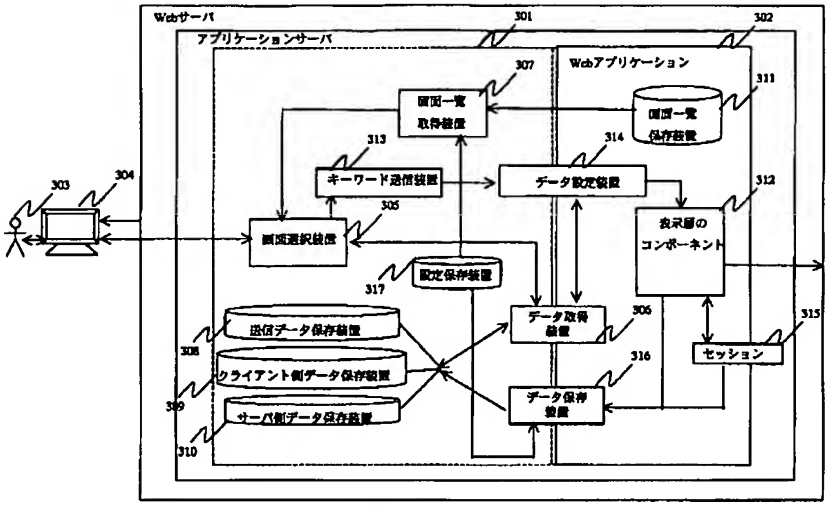
データファイルの構造の一例

【図２】



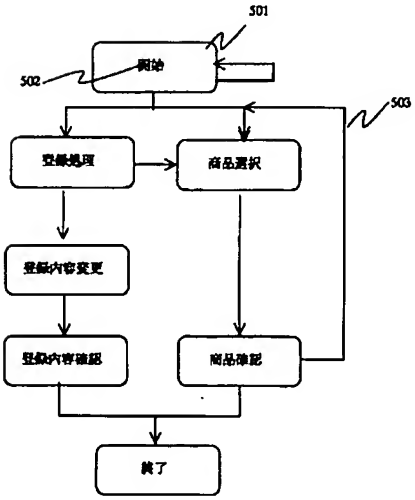
キーワードとデータとデータの種類の対応表(開始画面の場合)の一例

【図３】



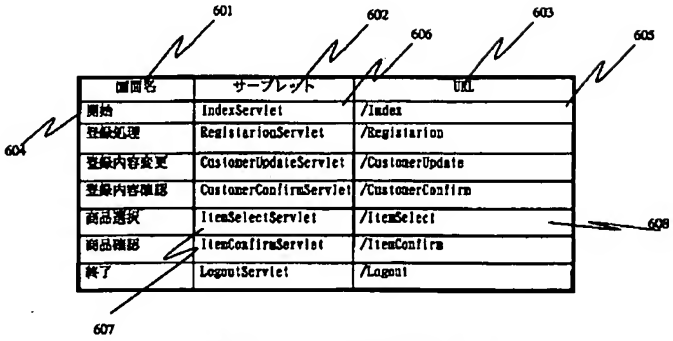
本発明の実施形態の一例

【図５】



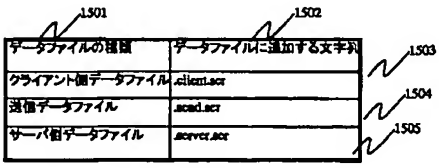
Webアプリケーションでの画面遷移の一例

【図６】



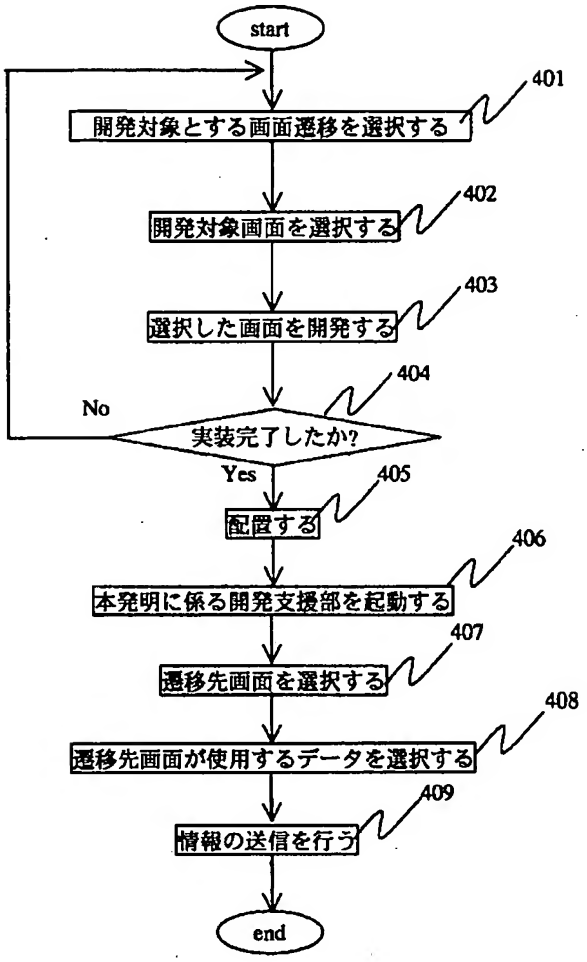
画面名とサーブレットとURLの対応表の一例

【図１５】



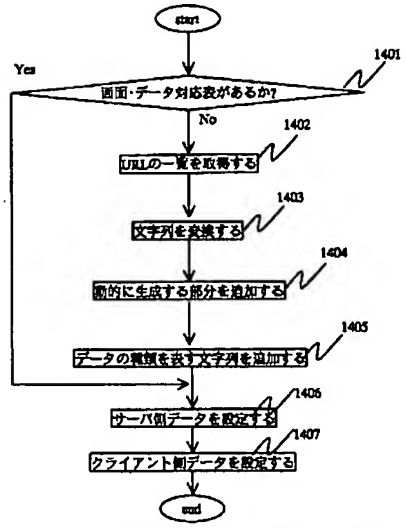
データファイルの種類とそれを表す文字列との対応表の一例

【図４】



本発明を用いてWebアプリケーション開発を行った場合のフロー図の一例

【図１４】



URL・データ対応表生成とデータ設定のフローの一例

【図９】

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://java.sun.com/j2ee/dtds/web-app_2_3.dtd">
<web-app>
  <servlet>
    <servlet-name>indexServlet</servlet-name>
    <servlet-class>IndexServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>indexServlet</servlet-name>
    <url-pattern>/Index</url-pattern>
  </servlet-mapping>
</web-app>
```

901 902 903 904 905 906 908 907 909 910

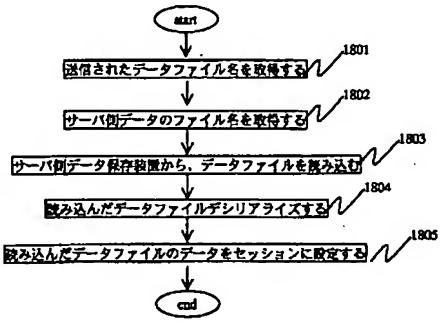
web.xmlの一例 (抜粋)

【図７】

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class BaseServlet extends HttpServlet{
        public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
                Hashtable clientData  = getClientData( req );
                Hashtable sendData  = getSendData( req );
                HttpSession sessionData  = getSessionData( req );
                String htmlStream =
                doBusiness( req, res,sessionData, clientData, sendData );

                setClientData( clientData );
                setSendData( sendData );
                setSessionData( sessionData );
                PrintWriter result;
                res.setContentType("text/html");
                result = res.getWriter ();
                result.println( htmlStream );
        }
}
```

サーブレット(スーパークラス)の実装の一例

【図１８】



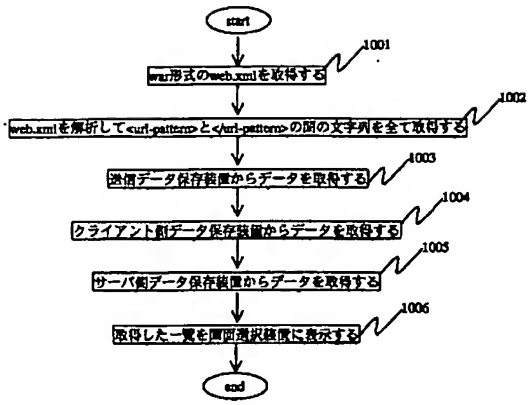
セッションに値を設定する処理のフローの一例

【図８】

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class IndexServlet extends BaseServlet{

        public String doBusiness(HttpServletRequest req, HttpServletResponse res, HttpSession sessionData, Hashtable clientData, Hashtable sendData)
                        throws ServletException, IOException {
                        String userID = (String)clientData.get( "USERID" );
                        String userName = getUserName( userID );
                        sendData.put( "USERNAME",  userName );
                        sessionData.put( "USERNAME",  userName );
                        sessionData.put( "USERID",  userID );
                        String htmlStream = "<html><head><title>Your Name</title></head><body><BR> Your Name :"
                                        + userName
                                        +"</body></html>";

                        return htmlStream;
        }
}
```

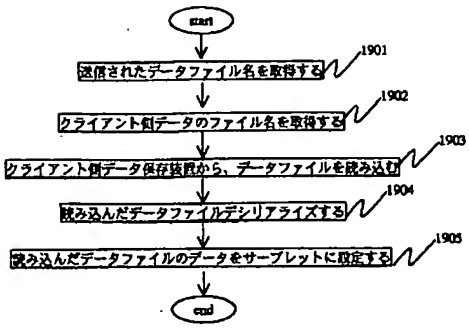
サーブレット(サブクラス)の実装の一例(IndexServlet)

【図１０】



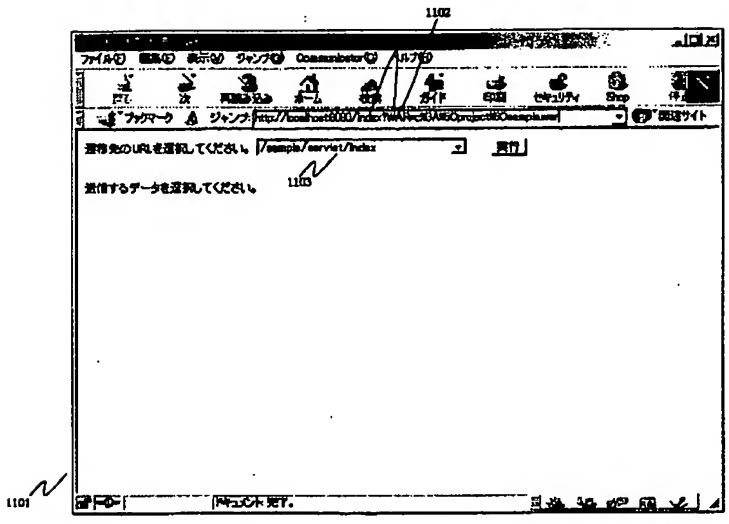
Webアプリケーションが公開しているURL一覧とデータファイルの一覧を取得し表示する処理のフローの一例

【図１９】



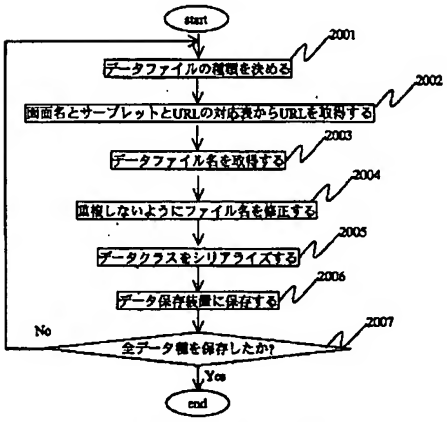
クライアント側データを設定する処理のフローの一例

【図１１】



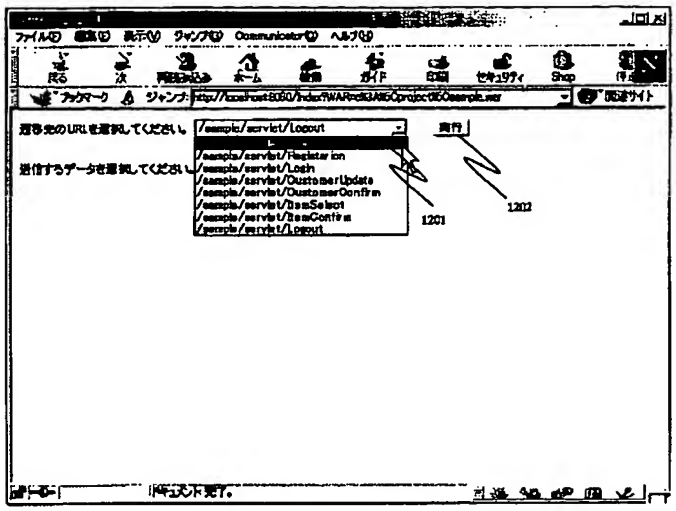
本発明に係る開発支援部を始めて起動したときの画面の一例

【図２０】



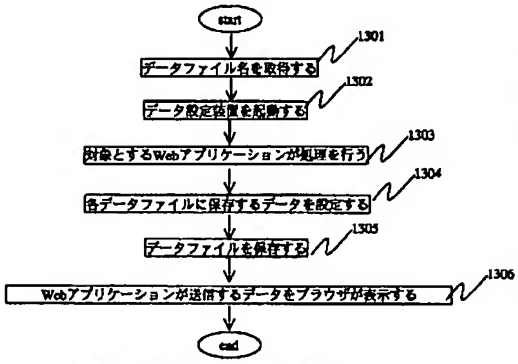
データを保存する処理のフローの一例

【図１２】



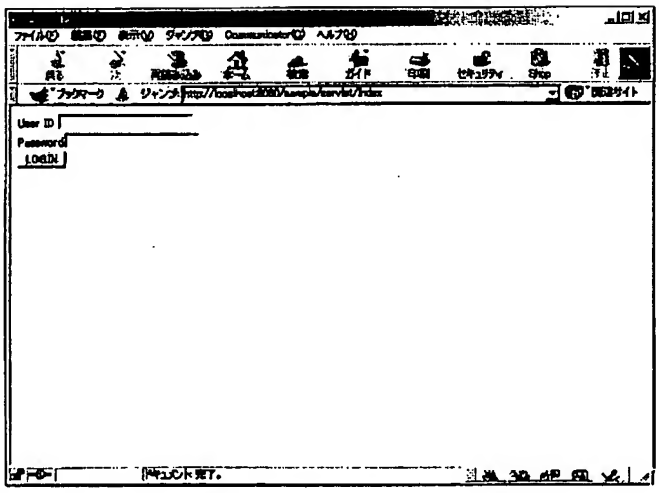
URLを選択している画面の一例

【図１３】



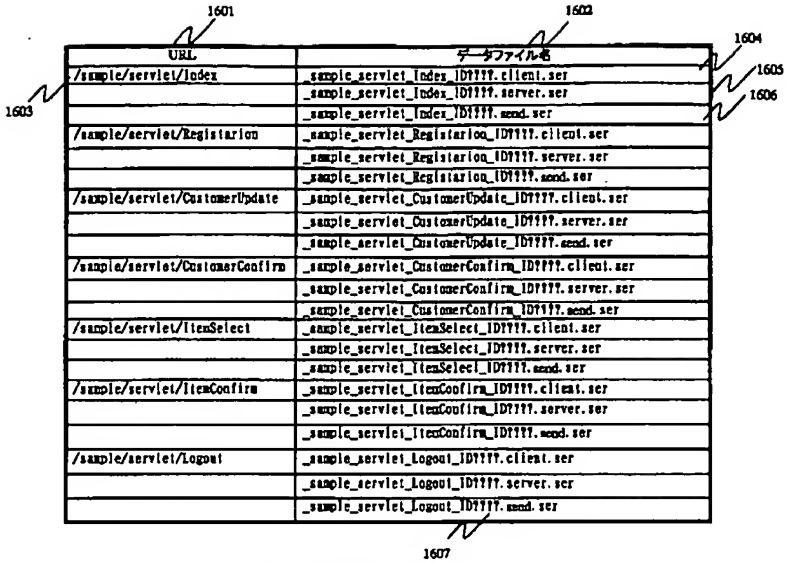
開発支援部が画面を表示するまでの処理のフローの一例

【図２１】



サーバから返却されたhtml形式データを表示した一例

【図１６】



URL	データファイル名
/sample/servlet/Index	_sample_servlet_Index_ID????.client.ser
	_sample_servlet_Index_ID????.server.ser
	_sample_servlet_Index_ID????.send.ser
/sample/servlet/Registarion	_sample_servlet_Registarion_ID????.client.ser
	_sample_servlet_Registarion_ID????.server.ser
	_sample_servlet_Registarion_ID????.send.ser
/sample/servlet/CustomerUpdate	_sample_servlet_CustomerUpdate_ID????.client.ser
	_sample_servlet_CustomerUpdate_ID????.server.ser
	_sample_servlet_CustomerUpdate_ID????.send.ser
/sample/servlet/CustomerConfirm	_sample_servlet_CustomerConfirm_ID????.client.ser
	_sample_servlet_CustomerConfirm_ID????.server.ser
	_sample_servlet_CustomerConfirm_ID????.send.ser
/sample/servlet/ItemSelect	_sample_servlet_ItemSelect_ID????.client.ser
	_sample_servlet_ItemSelect_ID????.server.ser
	_sample_servlet_ItemSelect_ID????.send.ser
/sample/servlet/ItemConfirm	_sample_servlet_ItemConfirm_ID????.client.ser
	_sample_servlet_ItemConfirm_ID????.server.ser
	_sample_servlet_ItemConfirm_ID????.send.ser
/sample/servlet/Logout	_sample_servlet_Logout_ID????.client.ser
	_sample_servlet_Logout_ID????.server.ser
	_sample_servlet_Logout_ID????.send.ser

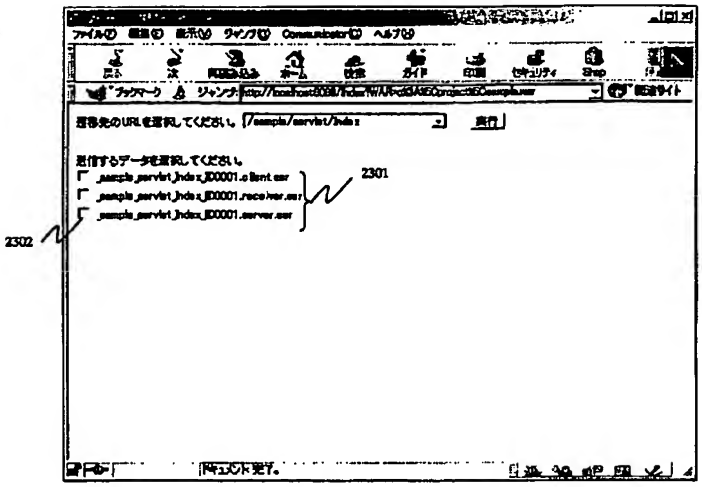
URL･データ対応表の一例

【図２２】

```
import javax.servlet.*;
import javax.servlet.http.*;                                                    2201

public class ItemSelectServlet extends BaseServlet{
        public String doBusiness(HttpServletRequest req, HttpServletResponse res, HttpSession sessionData, Hashtable clientData, Hashtable sendData)
                throws ServletException, IOException {
                String userName = (String)sessionData( "USERNAME" );                    2202
                String htmlStream = null;
                if( userName == null ){                                         2203
                        return htmlStream =  "<html><head><title>Please Login</title></head><body><BR>Please Login</body></html>";
                }
                String itemList = getItemList();                                2204
                sessionData.put( "USERNAME",  userName );                       2205
                String htmlStream = "<html><head><title>Your Name</title></head><body><BR> Your Name :"
                                + userName                                      2206
                                + itemList
                                +"</body></html>";
                return htmlStream;
        }
}
```

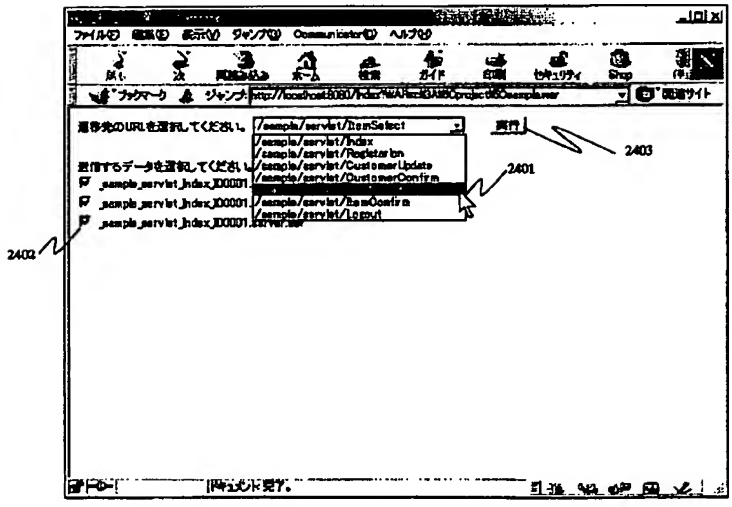
サーブレット(サブクラス)の実装の一例(ItemSelectServlet)

【図２３】



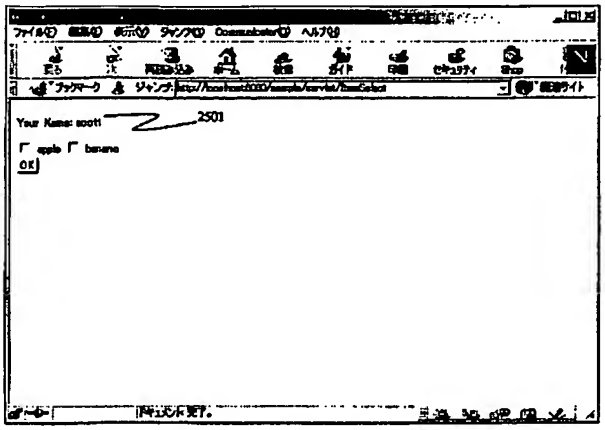
二回目以降に実行されたときの画面の一例

【図２４】



URLとデータファイルを選択している画面の一例

【図２５】



Webアプリケーションの処理によってブラウザに表示された画面の一例